2008

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

Keith Ronald Schreck
San Jose State University

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

San Jose State University

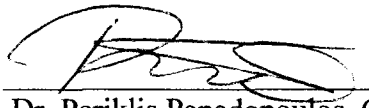
The undersigned Faculty Committee Approves

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

of

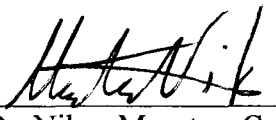
Keith Ronald Schreck

APPROVED FOR THE DEPARTMENT OF MECHANICAL AND AEROSPACE ENGINNERING



4/1/08

Dr. Periklis Papadopoulos, Committee Chair Date



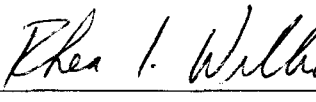
4-1-08

Dr. Nikos Mourtos, Committee Member Date

4/04/08

Mr. Nikola Djordjevic, Committee Member Date
Lockheed Martin Space Systems Company

APPROVED FOR THE UNIVERSITY



06/17/08

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 pf 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

LIST OF FIGURES

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.

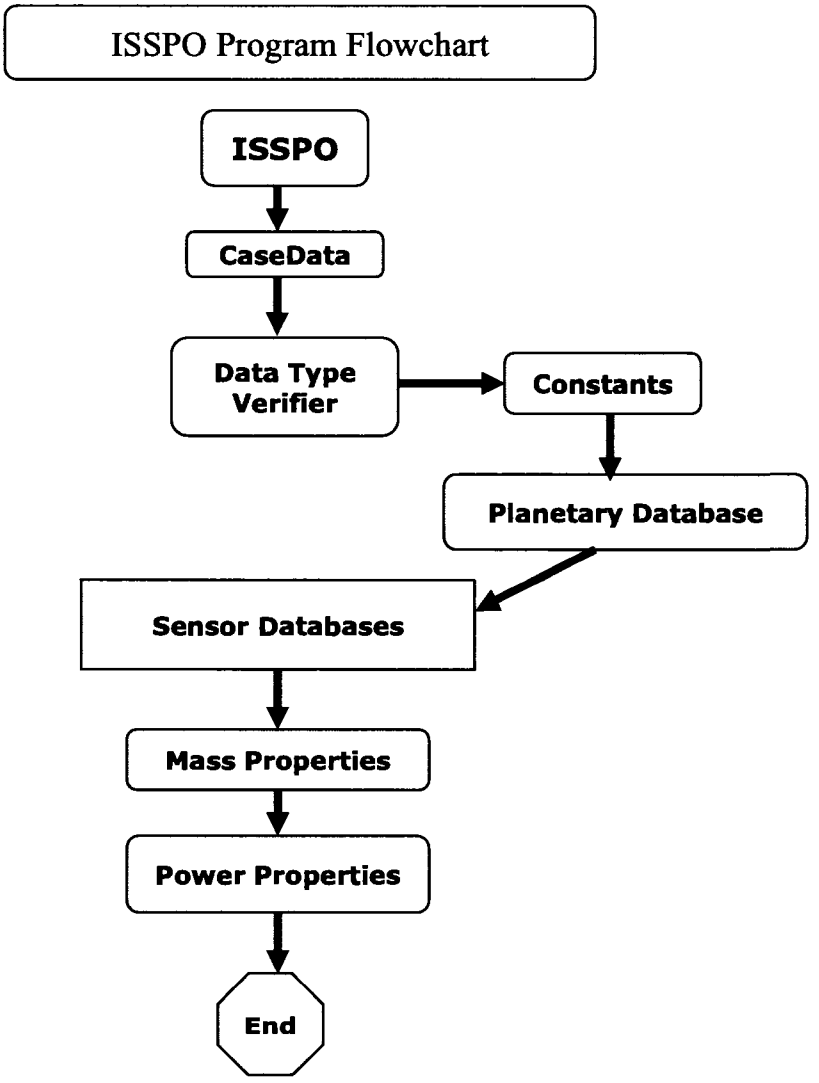


Figure 1 ISSPO Tool Flowchart Diagram

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 aide 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_2) and the second is the numerical percent composition of the planet's atmosphere (e.g., Earth 'N_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.

AC Field Sensor Database Properties	
Common Sensor Properties	
Sensor Type	Number of Axis
Low Magnetic Field Limit	High Magnetic Field Limit
Operational Frequency Range	Accuracy
Sampling Time Interval	Sensitivity
Voltage Requirements	
Unit System Specific Properties	
Sensor Operational Temperature Range	
Electronics Operational Temperature Range	
Sensor Dimensions	Electronics Dimensions
Sensor Weight	Electronics Weight
Sensor Comments	

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.

Accelerometer Database Properties	
Common Sensor Properties	
Sensor Type	Dynamic Range
Sensitivity	Operational Frequency Range
Linearity	Resonance Frequency
Shock Limits	Circuit Configuration
Current Range	Voltage Requirements
Electrical Signal Noise	
Unit System Specific Properties	
Sensor Temperature Range	Sensor Weight
Sensor Dimensions	
Temperature Sensitivity Correction	

Table 3 Acceleration Sensor Input File Format

1st Column	2nd Column	3rd Column
'ACCELERATION'	'LOW'	'Voltage'
	'MEDIUM'	'Const Current'
	'HIGH'	'Self Generating'
	'IMPACT'	
	'BALLISTIC'	

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 4 Acoustic Sensor Data recorded in ISSPO Tool.

Acoustic Sensor Database Properties	
Common Sensor Properties	
Sensor Type	Dynamic Range
Frequency Range	Sensitivity
Resonance	Inherent Noise
Clipping Limit	Voltage Range
Current Range	Stability
Unit System Specific Properties	
Operational Temperature Range	
Storage Temperature Range	
Temperature Coefficient	Pressure Coefficient
Sensor Dimensions	Weight

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 5 Acoustic Sensor Input File Format

1st Column	2nd Column
'ACOUSTICS'	'SENSOR'
	'ARRAY'
	'VELOCITY'

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 Wikepedia [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.

$$\rho = \frac{P}{RT} \qquad (1)$$

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 6 Density Sensor Data recorded in ISSPO Tool.

Density Sensor Database Properties	
Unit System Specific Properties	
Sensor Type	Density Range
Accuracy	Repeatability
Temperature Range	Maximum Pressure
Weight	Dimensions
Voltage Requirements	Current Requirements
Vibration Limits	Tolerance
Viscosity	Flowrates

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 7 Density Sensor Input File Format

1st Column	ANY Subsequent Column	Next Column
"ATMOSPHERIC'	"DENSITY'	"GAS'
		"LIQUID'
"ATMOSPHERIC'		"GAS'

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.

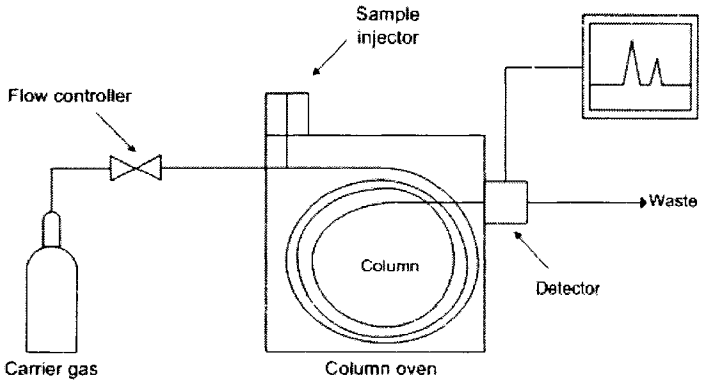


Figure 2 Gas Chromatograph Reference Diagram [13]
Reprinted from http://en.wikipedia.org/wiki/Gas_chromatography

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.

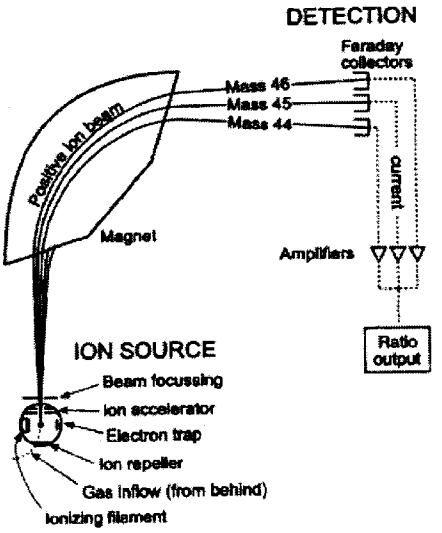


Figure 3 Mass Spectrometer Reference Diagram [14]
Reprinted from http://en.wikipedia.org/wiki/Mass_spectrometry

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

1st Column	2nd Column	3rd Column
"GAS ANALYSIS"	"MASS-CHARGE"	"LOW"
		"MEDIUM"
		"HIGH"
"GAS ANALYSIS"	"WAVELENGTH"	

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.

GCMS Sensor Database Properties	
Unit System Specific Properties	
Sensor Type	Wavelength Range
Spectral Width	Resolution
Scan Time	Detector Size
Weight	Dimensions
Voltage	Sample Size

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 10 GCMS – Mass-Charge Sensor Data recorded in ISSPO Tool.

GCMS Sensor Database Properties	
Unit System Specific Properties	
Sensor Type	Mass-Charge Range
Mass Filter Material	Detector Type
Dynamic Scan Range	Resolution
Sensitivity	Minimum Partial Pressure
Number of Ion Sources	Construction Material
Filament Material	Field Range
Electron Energy	Ion Energy
Focus Voltage	Electron Current
Dimensions	Weight
Mass Stability – Mass	Mass Stability – Time
Input Voltage	Voltage Type
Current	Data Rate
Power Requirements – Min/Max/Mean	

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.

Humidity Sensor Database Properties	
Unit System Specific Properties	
Sensor Type	Relative Humidity Range
Sensitivity	Hysteresis
Response Time	Stability
Temperature Range	Frequency Range
Dimensions	Weight

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.

Inclinometer Sensor Database Properties	
Unit System Specific Properties	
Sensor Type	Inclination Range
Output Voltage Range	Resolution
Sensitivity	Non-linearity
Input Voltage	Response Time
Bandwidth	Operating Temperatures
Storage Temperatures	Dimensions
Weight	

Table 13 Inclinometer Sensor Input File Format

1st Column	2nd Column
"INCLINATION"	"LOW"
	"MEDIUM"
	"HIGH"

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.

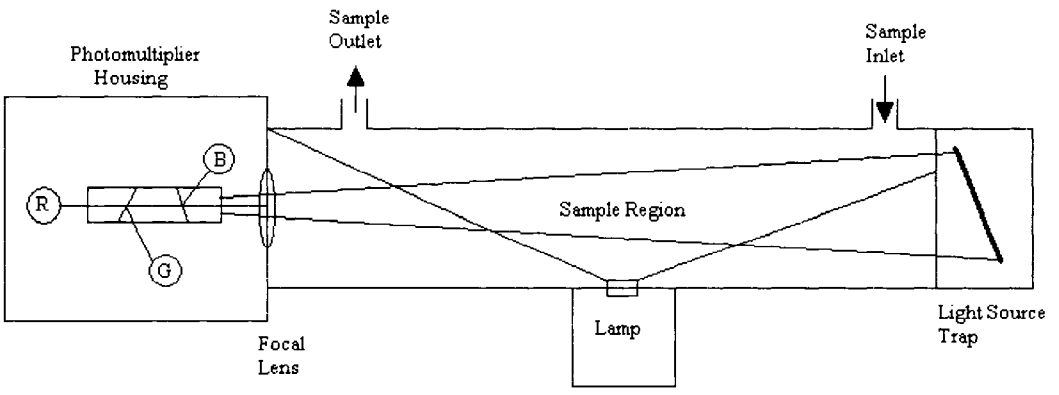


Figure 4. Simplified Nephelometer Reference Diagram

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.

Nephelometer Sensor Database Properties	
Nephelometer Properties	
Sensor Type	Operating Wavelengths
Sensor Bandwidth	Weight
Power Requirements	Voltage
Dimensions	Integration Time
Sensor drift	Response time
Sample Flow Rates	Operating Temperature
Relative Humidity Limits	
Power Supply Properties	
Weight	Power Supply
Voltage	Power Supply Dimensions

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 Data recorded in ISSPO Tool.

Optical Imaging Sensor Database Properties	
Sensor Type	Optical Range Type
CCD Array Dimensions	Pixel Size
Image Area	Signal Read Noise
Full Well Capacity	Gain
Linearity	ADC Dynamic Range
Readout Rates	Readout Time
Frame Rate	Dimensions
Operating Temperature	Cooling Temperature
Cooling Method	ADC Dynamic Range
Weight	

Table 16 Optical Sensor Input File Format

1st Column	2nd Column	3rd Column	>4th Column
"OPTICS"	"CAMERA"	"LOW"	"X-RAY"
	"ARRAY"	"MEDIUM"	"VISUAL"
		"HIGH"	"UV"
			"NIR"
			"MICRO"

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 17 Pressure Sensor Data recorded in ISSPO Tool.

Pressure Sensor Database Properties	
Sensor Type	Accuracy
Stability	Over Pressure Rating
Burst Pressure Rating	Power Requirements
Number of Life Cycles	Pressure Range
Shock Limits	Impulse Time
Operating Frequency Range	Humidity Range
Operating Temperature	Storage Temperature
Voltage Requirements	Current Requirements
Sensor Material Construction	Dimensions
Weight	

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 suns 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 suns radiation and any incident cosmic radiation. Certain types of radiation can be effectively blocked

by various mediums. Mapping of a planets 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 18 Radiation Sensor Input File Format

1st Column	>2nd Column
"RADIATION"	"Charged Particle"
	"Alpha"
	"Beta"
	"Gamma"
	"X-Ray"

Table 19 Radiation Sensor Data recorded in ISSPO Tool.

Radiation Sensor Database Properties	
Sensor Type	
Proton Energy Range	Number of Proton Detectors
Electron Energy Range	Number of Electron Detectors
Alpha Particle Energy Range	Number of Alpha Particle Detectors
Beta Particle Energy Range	Number of Beta Particle Detectors
Gamma Particle Energy Range	Number of Gamma Particle Detectors
X-Ray Energy Range	Number of X-Ray Detectors
Particle Count Rate	G-Factor
Weight	Dimensions
Power Requirements	

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.

Refraction Sensor Database Properties	
Sensor Type	Refraction Index Range
Resolution	Sensor Drift
Accuracy	Weight
Voltage	Current
Power	Dimensions
Operating Temperatures	

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.

Table 21 Temperature Sensor Input File Format

1st Column	ANY Subsequent Column	Next Column
"ATMOSPHERIC"	"TEMPERATURE"	"VOLTAGE"
		"RESIATANCE"
"ATMOSPHERIC"		"VOLTAGE"

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 22 Temperature Sensor Data recorded in ISSPO Tool.

Temperature Sensor Database Properties	
Thermocouple Properties	
Sensor Type	Cold Temperature Limit
Hot Temperature Limit	Error Tolerance Percent (%)
Error Tolerance Temperature (deg)	Positive Lead Material
Negative Lead Material	Minimum EMF Voltage
Maximum EMF Voltage	Usage/Restriction Comments
Insulator Properties	
Insulator Name	Cold Limit
Hot Limit	Abrasion Resistance
Flexibility	Water Submersion
Resistance to Solvents	Resistance to Acids
Resistance to Bases	Resistance to Flame
Resistance to Humidity	Material Construction: Overall
Material Construction: Conductors	

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.

Table 23 Wind Velocity Sensor Input File Format

1st Column	ANY Subsequent Column	Next Column
"ATMOSPHERIC"	"WIND VELOCITY"	"ANEMOMETER"
		"DOPPLER"
"ATMOSPHERIC"		"ANEMOMETER"

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 24 Wind Velocity Anemometer Sensor Data recorded in ISSPO Tool.

Wind Velocity Sensor Database Properties	
Sensor Type	Wind Velocity
Accuracy	Volumetric Flow Rate Ability
Probe Temperature Range	Instrument Temperature Range
Power Requirements	Weight
Electronics Dimensions	Probe Dimensions

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 25 Wind Velocity Doppler Sensor Data recorded in ISSPO Tool.

Wind Velocity Sensor Database Properties	
Sensor Type	Output Frequency
Output Level	Aging Data
Spectral Purity – Harmonics/Non-Harmonics	
Accuracy	Stability
Warm up Time	System Voltage
Operating Temperature Range	Storage Temperature Range
Temperature Sensitivity	Orientation Sensitivity
Pressure Sensitivity	Power Requirements
Weight	Dimensions

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 26 Digital Signal Processing Input File Format

1st Column	2nd Column	3rd Column
"DATA ANALYSIS"	"LOW"	Number of devices
	"MEDIUM"	
	"HIGH"	

Table 27 Digital Signal Processing Data recorded in ISSPO Tool.

Digital Signal Processor Database Properties	
Sensor Type	CPU Clock Speed
CPU Cores	Data Memory
Program Memory	Computational Units
Cycle Time	External Interrupts
System Voltage	Operating Temperature Range
Storage Temperature Range	CPU Power Requirement
Weight	Voltage
Current	Dimensions

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 in 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 shownin 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.

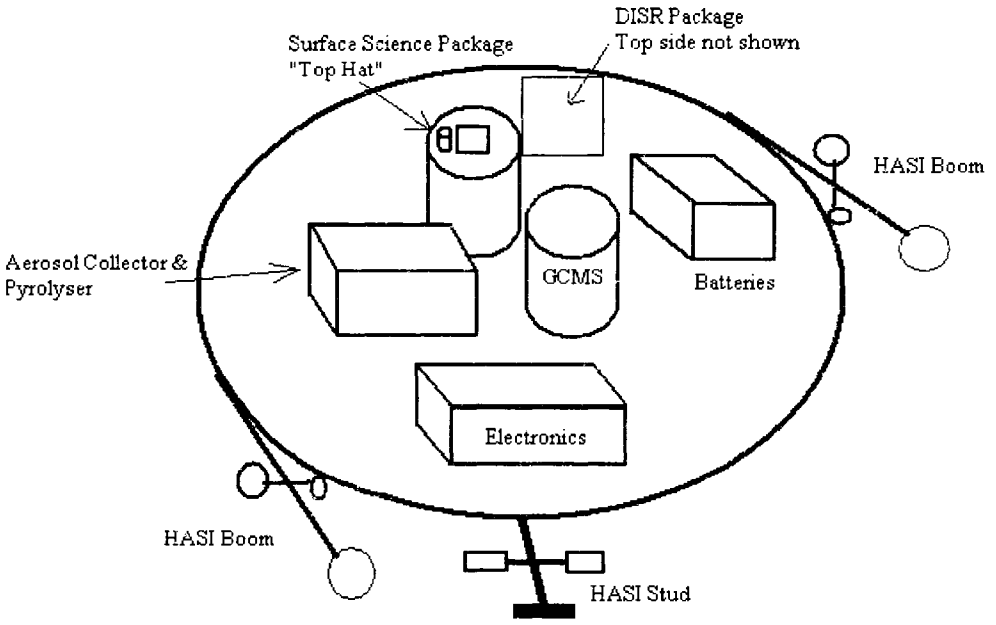


Figure 5 Schematic Diagram of Huygens Probe Instrument Layout

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

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

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

Descent Imager / Spectral Radiometer Properties	
Instrument Overview:	
CCD Detector Array	Two Linear IR detector Arrays
Surface Science Lamp	Sun Sensor
In-flight Calibration System	Hardware and software data compression systems
Flexible data collection software	Three frame imagers
Upward and downward-looking IR spectrometer	Upward and downward-looking visible spectrometer
Upward and downward-looking violet photometer	4-channel solar aureole camera
Payload Properties:	
Mass = 8.07 kg	Power (typical/peak) = 13 / 70 W
Energy = 42 Wh	Data Rate = 4800 bits/sec

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.

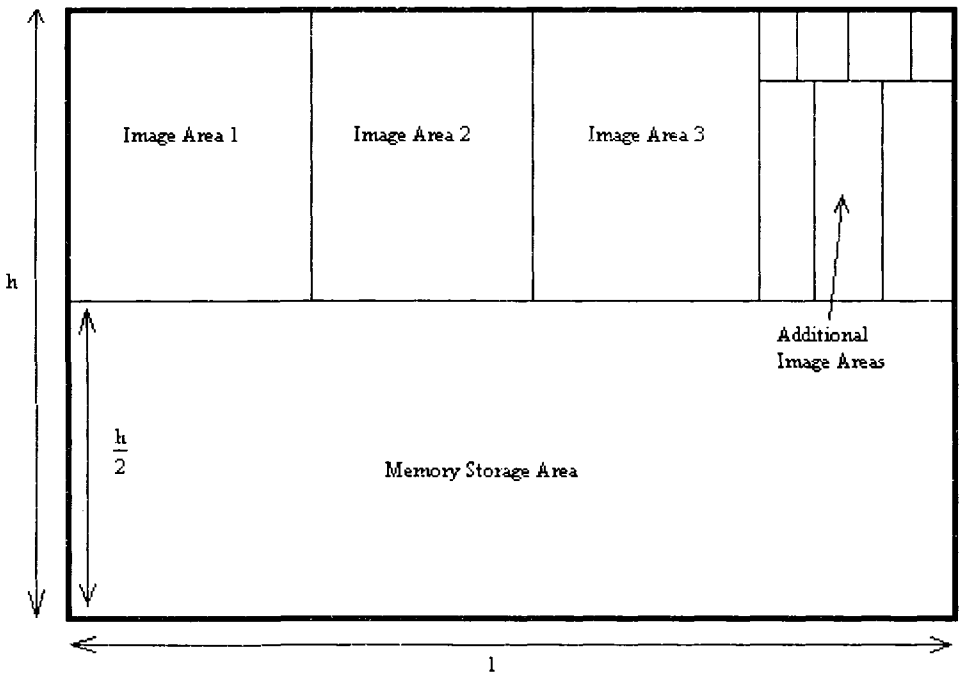


Figure 6 Schematic Diagram of CCD Array Image Layout and Binning

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

1st Column	2nd Column	3rd Column	>4th Column
"OPTICS"	"ARRAY"	"LOW"	"VISUAL"
			"UV"
			"NIR"

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 31 ISSPO System Architecture DISR CCD Array

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
Dimensions - Length:	73.15 mm
Width:	52.83 mm
Height:	6.10 mm

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:		
Mass (g)	1898	radiation shielding: 150 g
Dimensions (mm)	170 × 117 × 119	L × W × H
DC power:		
Warm-up power (W)	< 18.4	< 30 min
DC consumption (mA)	< 675	System limit: 0.7 A
Energy (Wh)		worst case (minimum temp)
Frequency Parameters:		
Output frequency (MHz)	10	± 0.1 Hz
Frequency long term drift	1.4×10^{-9}	$\delta f_0/f_0$ over 3 hours
Allen Variation	3×10^{-11}	$\tau = 1$ s
	6×10^{-12}	$\tau = 10$ s

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

1st Column	2nd Column	3rd Column
"ATMOSPHERIC"	"WIND VELOCITY"	"DOPPLER"

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:

$$\Delta f = -\frac{f}{c}\Delta V \qquad (2)$$

$$\Delta V = \left(\vec{V}_p - \vec{V}_o\right) \cdot \vec{R}_{op} \qquad (3)$$

where p and o are the probe and orbiter elements, respectively. And R_{op} is a unit vector path from the orbiter to the probe defined as:

$$\vec{R}_{op} = \frac{\vec{R}_p - \vec{R}_o}{\left|\vec{R}_p - \vec{R}_o\right|} \qquad (4)$$

The ΔV 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.

$$\vec{R}_{op} \cdot \vec{V}_p = V_1 + V_2 + V_3 + V_4 \qquad (5)$$

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 + vertical wind w (positive upwards)

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

$$\vec{R}_{op} \cdot \vec{V}_o = V_5 \qquad (6)$$

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 34 ISSPO DWE Sensor Configuration

ISSPO DWE Sensor System Architecture Result		
Model : Symmetricom 8130A		
Physical Properties:		
Mass (g)	900	
Dimensions (mm)	102.6 × 74.1 × 72.8	L × W × H
DC power:		
Warm-up power (W)	<35.0	< 10 min
DC consumption (mA)	1094	At 32 V DC input
Energy (Wh)		worst case (minimum temp)
Frequency Parameters:		
Output frequency (MHz)	10	
Frequency long term drift	$<5.0 \times 10^{-11}$	After 1 month
Allen Variation	3×10^{-11}	$\tau = 1$ s
	3×10^{-11}	$\tau = 10$ 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.

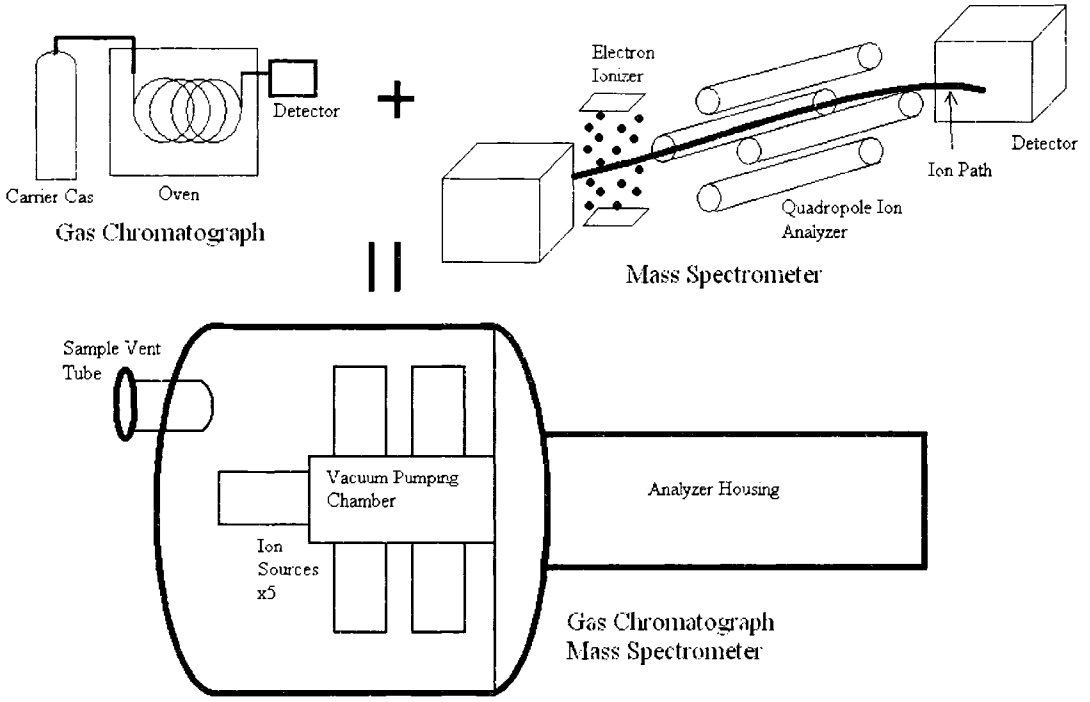


Figure 7 Schematic Diagram of Huygens GCMS Sensor

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

1st Column	2nd Column	3rd Column
"GAS ANALYSIS"	"MASS-CHARGE"	"MEDIUM"

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

ISSPO GCMS System Architecture Design Result		
Model : GCMS - Huygens		
Physical Properties:		
Mass (kg)	17.30	
Dimensions (mm)	470 × 198 × 198	L × W × H
DC power:		
Typical Power (W)	28	After 36 min warm-up period
Average Power (W)	41	At 28 V DC input
Peak Power (W)	71	
Sensing Parameters:		
Mass – Charge Range	2-141	amu
Number of Ion Sources	5	units
Ion Source Charge Field Range	1.00	W degas
	16.00	W degas

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.

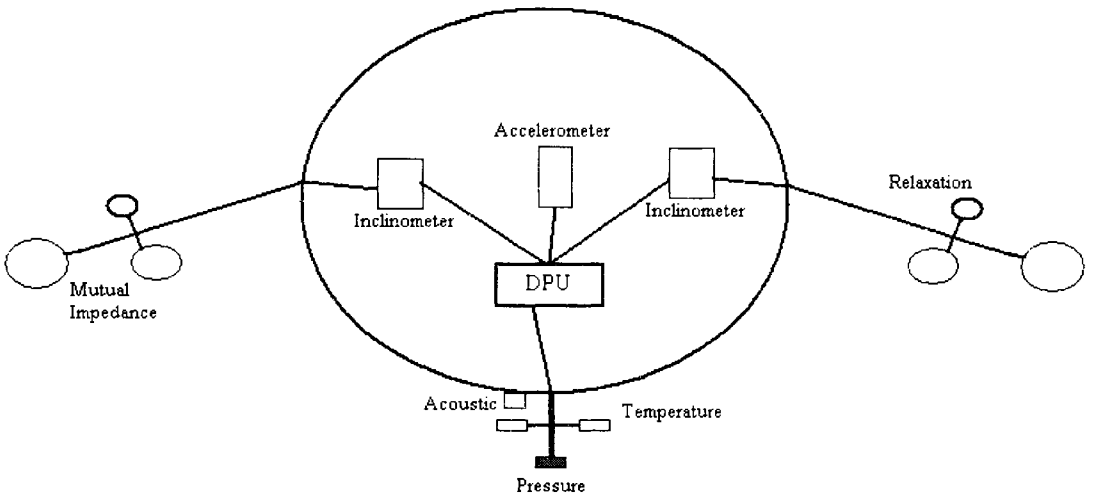


Figure 8 Schematic Diagram of Huygens HASI Sensor Package

A diagram of the different sensor components contained in the HASI package in 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 37 Huygens HASI Science Package Objectives

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

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 38 HASI Sensor Package Summary Results

HASI Sensor Package Components		
Sensor Package	**Flight Unit**	**ISSPO Results**
Accelerometer – 3 axis servo	Sundstrand QA-2000-030	QA2000-030
Accelerometer - piezo-resistive	Endevco 7264A-2000T	Endevco 7264A-2000T
Pressure Profile Instrument	Vaisala - Barocap	Series 48-0025
Temperature Sensor	dual element platinum resistance thermometers Rosemount Aerospace Inc.	Goodrich Model 0146MD
PWA – Acoustic Sensor	Kulite CT-190M	Kulite CT-190M
PWA – Digital Signal Processor(FFT)	Analog Devices ADSP-2100A	ADSP-2100A

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:

$$\rho = -\frac{2ma}{C_D A V^2} \qquad (7)$$

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.

Table 39 ISSPO HASI Accelerometer Sensor Configuration

ISSPO HASI Accelerometer Sensor Package Results		
Model :	QA-2000-030	Endevco 7264A-2000T
Physical Properties:		
Mass (g)	71	1.0
Dimensions (mm)	27.18 × 25.53 × 25.53	5.08 × 10.16 × 10.16
Sensing Parameters:		
Acceleration Range (g's)	+/- 60	+/- 2000
Transverse Sensitivity	3.0 %	1 %
Repeatability	≤ 160 μg	0.2 g

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

ISSPO HASI PPI Sensor Package Comparison		
Model :	Barocap – Flight Unit	ASCO 48-0025
Physical Properties:		
Dimensions (mm)	70.00 × 50.0 × 17.00	48.56 × 22.23 × 22.23
Sensing Parameters:		
Pressure Range (g's)	0 – 2000 mbar	0 – 1724 mbar
Accuracy	1.0 %	0.5 %

3. Temperature Sensors

The flight unit temperature sensors carried as part of the HASI sensor package features a coarse and fine sensor design. The designs 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.

$$T_{corr} = T_{meas} - r\left[\frac{V_r^2}{2C_p}\right] \qquad (8)$$

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 41 ISSPO HASI Temperature Sensor Input File Format

1st Column	2nd Column	3rd Column
"ATMOSPHERIC"	"TEMPERATURE"	"RESISTANCE"

Table 42 ISSPO HASI Temperature Sensor Configuration

ISSPO HASI Temperature Sensor Package Results		
Model : M-0146MD		
Physical Properties:		
Mass (g)	0.35	
Dimensions (mm)	45.72 × 1.524 × 1.524	L × W × H
Sensing Parameters:		
Temperature Range	-269 – 400	deg C
Stability	0.03	%
Repeatability	≤ 0.1	deg C

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

1st Column	2nd Column
"ACOUSTICS"	"SENSOR"

Table 44 ISSPO HASI Acoustic Sensor Configuration

ISSPO HASI Acoustic Sensor Package Results		
Model : Kulite CT-190M		
Physical Properties:		
Mass (g)	4.0	
Dimensions (mm)	34.3 × 9.5 × 9.5	L × W × H
Sensing Parameters:		
Acoustic Range	56.8 – 135	dB_{SPL}
Resonance	500	kHz
Temperature Range	-195.5 – 37.0	deg C

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 45 ISSPO DSP Component Input File Format

1st Column	2nd Column	3rd Column
"DATA ANALYSIS"	"LOW"	4

Table 46 ISSPO HASI DSP Component Configuration

ISSPO HASI DSP Sensor Package Results		
Model : Analog Devices ADSP-2100		
Physical Properties:		
Output Power (mW)	790.0	
Dimensions (mm)	33.83 × 33.83 × 9.12	L × W × H
Operating Parameters:		
CPU Operating Speed	8.192	MHz
Data Memory	16 K-Words	16 Bins
Program Memory	32 K-Words	24 Bins
Temperature Range	-55.0 – 125.0	deg C

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 47 Huygens Surface Science Package Scientific Objectives

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

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.

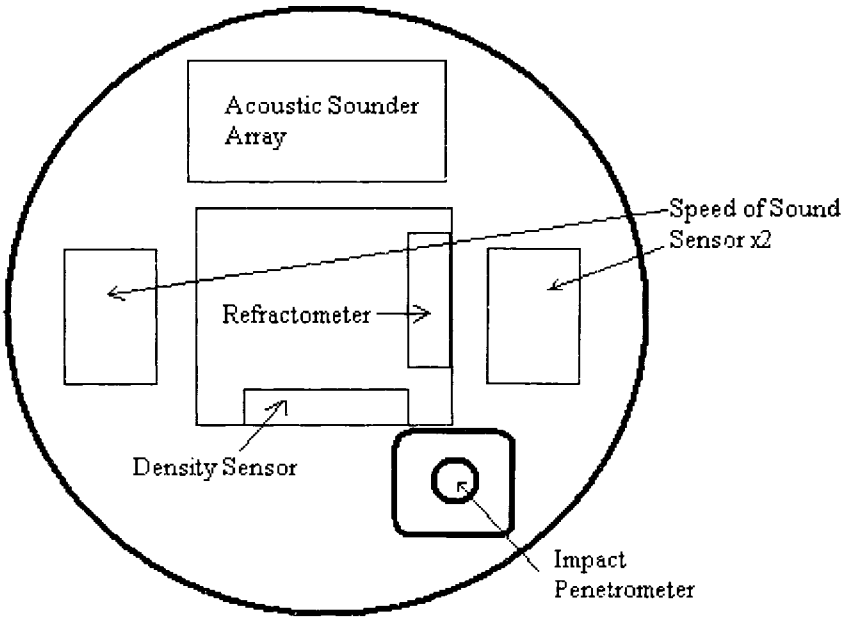


Figure 9 Schematic Diagram of Huygens SSP Top Hat Sensor Cavity

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.

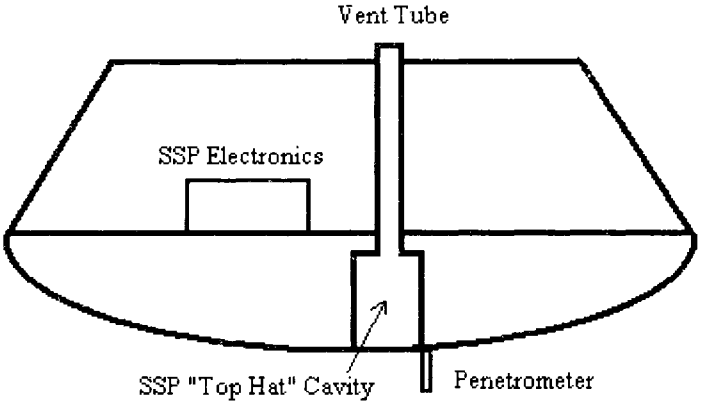


Figure 10 Schematic Diagram of Huygens SSP Component Layout

Table 48 ISSPO - Huygens SSP Sensor Component Comparison

SSP Sensor Components		
Sensor	**Flight Unit**	**ISSPO Results**
Accelerometer – Impact penetrometer ACC-E	Piezoelectric Ceramic PZT-5A	N/A
Accelerometer - Impact accelerometer ACC-I	Endevco 2271AM20 0 – 100 g's	Endevco 2271AM20
Tilt Sensor TIL	Spectron L-211U +/- 60°	Spectron L-211U
Temperature Sensor THP	Hot Wire 65 – 100 K	M-0146MD
Velocity of Sound API-V	Piezoelectric Transducers – 2 150 – 2000 m/s	Physical Acoustics R80 Alpha
Acoustic Sounder API-S	Piezoelectric Transducers Array	N/A
Fluid Permittivity PER	Capacitance Sensor	N/A
Density of Fluid DEN	Archimedes Sensor 400 – 700 kg/m^3	N/A
Refractive Index REF	Critical Angle Refractometer 1.25 – 1.45	Huygens Design

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 49 ISSPO SSP Accelerometer Sensor Input File Format

1st Column	2nd Column	3rd Column
"ACCELERATION"	"IMPACT"	"Self Generating"

Table 50 ISSPO SSP Accelerometer Configuration

ISSPO SSP Accelerometer Sensor Package Results		
Model : Endevco 2271AM20		
Physical Properties:		
Dimensions (mm)	29.2 × 15.88 × 15.88	L × W × H
Operating Parameters:		
Sensing Range	0 – 10,000	g's
Transverse Sensitivity	3	%
Temperature Range	-269.0 – 260.0	deg C

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 51 ISSPO SSP Tilt Sensor Input File Format

1st Column	2nd Column
"INCLINATION"	"HIGH"

Table 52 ISSPO SSP Tilt Sensor Configuration

ISSPO SSP Tilt Sensor Package Results		
Model : Spectron L-211U		
Physical Properties:		
Dimensions (mm)	41.0 × 18.4 × 15.9	L × W × H
Operating Parameters:		
Tilt Range	+/- 60	Degrees
Resolution	0.0083	Degrees
Temperature Range	-54.0 – 124.0	deg C

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.

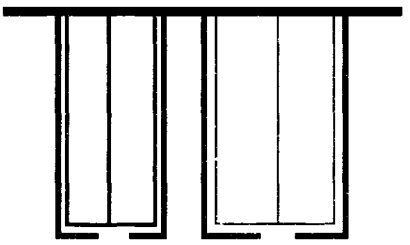


Figure 11 Schematic Diagram of Huygens SSP THP Sensor

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.

$$\Delta T = \frac{q}{4\pi\lambda}\ln\left(\frac{4\kappa t}{a^2 C}\right) \quad (9)$$

$$\lambda = \frac{q}{4\pi A} \quad (10)$$

$$\kappa = \left(\frac{a^2 C}{4}\right)\exp\left(\frac{B}{A}\right) \quad (11)$$

In Equations. 9 – 11, ΔT is the temperature rise in the wire, λ is the thermal conductivity of the medium, κ is the thermal diffusivity of the medium, q is the power per unit length of wire, α 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^g, 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 ΔT 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 53 ISSPO SSP THP Sensor Input File Format

1st Column	2nd Column	3rd Column
"ATMOSPHERIC"	"TEMPERATURE"	"RESISTANCE"

Table 54 ISSPO SSP THP Sensor Configuration

ISSPO HASI Temperature Sensor Package Results		
Model : M-0146MD		
Physical Properties:		
Mass (g)	0.35	
Dimensions (mm)	45.72 × 1.524 × 1.524	L × W × H
Sensing Parameters:		
Temperature Range	-269 – 400	deg C
Stability	0.03	%
Repeatability	≤ 0.1	deg C

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.

$$c = \sqrt{\frac{B}{\rho}} \qquad (12a)$$

$$c = \sqrt{\frac{\gamma RT}{M}} \qquad (12b)$$

In Equation 12a, B represents the Young's modulus of the medium, and Equation 12b is a variation of the Ideal Gas equation, where γ is the ratio of specific heats (~ 1for 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

ISSPO SSP Acoustic Sensor Package Results		
Model : Physical Acoustics R80 Alpha		
Physical Properties:		
Mass (g)	32	
Dimensions (mm)	19.0 × 19.0 × 21.4	L × W × H
Operating Parameters:		
Frequency Range	200 – 1000	kHz
Temperature Range	-54.0 – 124.0	deg C

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 ρ is the density, α is the molecular polarisability, N_A is Avogadro's number, ε_0 is the electrical permittivity of free space and ε_0 is the refractive index permittivity from the refractometer.

$$\alpha = \frac{3\varepsilon_0(\varepsilon_r - 1)}{\rho N_A(\varepsilon_r + 2)} \qquad (13)$$

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

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.

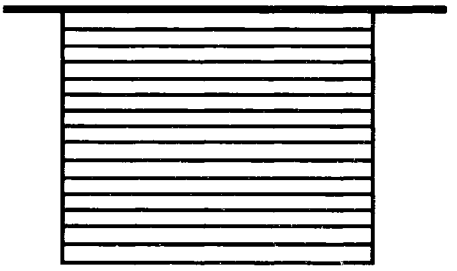


Figure 12 Schematic Diagram of Huygens SSP Permittivity Sensor Design

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 Θ_c of the prism. Rearranged the equation [85] for "s" is given as a function of the prism (μ_S) and fluid (μ_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.

$$s = r \times \left(\mu_L / \mu_S \right) \qquad (14)$$

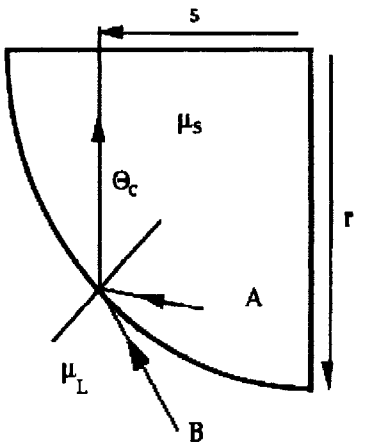


Figure 13 Schematic Diagram of a Refractometer Prism

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

ISSPO SSP Refractometer Sensor Package Results		
Model : Huygens - Ref		
Physical Properties:		
Dimensions (cm)	10.0 × 10.0 × 11.6i	L × W × H
Power (mW)	10	Photodiode array requirement
Operating Parameters:		
Refractive Index Range	1.25 – 1.45	Refractive Index Units
Accuracy	0.001	Refractive Index Units

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

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

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 59 Venus Atmosphere Concept Mission ISSPO Input Format

1st Column	2nd Column	3rd Column	4th Column	5th Column	6th Column
"ATMOSPHERE"	"WIND VELOCITY"	"ANEMOMETER"	"TEMPERATURE"	"VOLTAGE"	
"INCLINATION"	"MEDIUM"				
"ACCELERATION"	"MEDIUM"	"Voltage"			
"EM FIELD"					
"RADIATION"	"Charged Particle"				
"OPTICS"	"ARRAY"	"MEDIUM"	"VISUAL"	"UV"	"NIR"

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

ISSPO Venus Atmosphere Sensor Package Results					
SENSOR	**NAME**	**RANGE**	**MASS**	**VOLUME**	**POWER**
ATMOSPHERE – Temperature	K	- 200 – 1250 deg C	0	0	0
ATMOSPHERE - Pressure	13(C,U)2000P	0 – 13789.5 kPa	0	29.41 cm^3	0.030 W
ATMOSPHERE – Wind Velocities	FT 702	0 – 70 m/s	0.500 kg	0.0004 m^3	0
INCLINATION	L-212T	-45 – 45 deg	0	12.00 cm^3	0
ACCELERATION	MA15	-50 - 50 g's	0.1417 kg	47.72 cm^3	0.64 W
EM FIELD	TAM-1	0 – 1000 mG	1.080 kg	0.0016 m^3	0
RADIATION	LPD	0 – 250 MeV	7.00	0.0067 m^3	15 W
OPTICS	CCD 3041	UV VISUAL NIR	0	3.384 cm^3	0
SUMMARY			8.7217 kg	0.00871 m^3	15.77 W

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 very 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 fame 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 e 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

[1] Rodgers, David. "Short Course on In-Situ Instruments for Planetary Probes and Aerial Platforms," *Proceedings from 4th International Planetary Probe Workshop.* Pasadena, CA: IPPW, 2006.

[2] National Aeronautics and Space Administration. Advanced Concepts Office. Office of Space Access and Technology. "Technology Readiness Levels: A White Paper", Washington, DC: NASA, 1995.

[3] Grayzeck, Ed. *Planetary Fact Sheets.* National Space Science Data Center. URL: http://nssdc.gsfc.nasa.gov/planetary/planetfact.html [cited 1 December 2007].

[4] Magnetic Sciences. *Handheld Magnetometers, Gaussmeters, Teslameters, and EMF Meters.* URL: http://www.magneticsciences.com/GaussmetersAndAccessories.html [cited 5 December 2007].

[5] Macintyre Electronics Design Associates, Inc. *Satellite Magnetometers.* URL: http://meda-sam.meda.com/catalog/cat4_1.htm [cited 7 December 2007].

[6] Honeywell International. *Honeywell Magnetometry Solutions.* Solid State Electronics Center. URL: http://www.ssec.honeywell.com/magnetic/magnetometers.html [cited 3 December 2007].

[7] Honeywell International. *Honeywell.* Product Data Sheets. URL: http://www.sensotec.com/catpages.asp [cited 29 November 2007].

[8] Omega. *Dynamic Measurement.* Accelerometers and Vibration Transducers. URL: http://www.omega.com/toc_asp/subsectionSC.asp?subsection=K02&book=Pressure [cited 4 December 2007].

[9] Bruel & Kjaer. *Bruel & Kjaer.* Acoustic Transducers. URL: http://www.bksv.com/3024.asp (2007).

[10] PCB Group. *Precision Microphones and Accessories.* Acoustic Measurement Products. [Online]. http://www.pcb.com/Linked_Documents/Vibration/Mics_1006.pdf. [cited 1 December 2007].

[11] Wikipedia. *Densitometer.* from Wikipedia, The Free Encyclopedia. URL: http://en.wikipedia.org/wiki/Densitometer [cited 6 October 2007].

[12] Mobrey. *Downloads - Documentation & Software.* from Emerson Process. URL: http://www.mobrey.com/downloads/view_all.php?type=datasheets&classid=density [cited 6 October 2007].

[13] Wikipedia. *Gas-liquid chromatography.* from Wikipedia, The Free Encyclopedia. URL: http://en.wikipedia.org/wiki/Gas_chromatography [cited 27 January 2008].

[14] Wikipedia. *Mass Spectrometry.* from Wikipedia, The Free Encyclopedia. URL: http://en.wikipedia.org/wiki/Mass_spectrometry [cited 27 January 2008].

[15] Hitachi High Technologies America. *Spectrophotometers.* URL: http://www.hitachi-hta.com/pageloader.aspx?type=product&id=390&orgid=45&gclid=CMCQxI2pzpACFRpdagodVmMrXg [cited 27 January 2008].

[16] Horiba Jobin Yvon Inc. *Spectrometers: Monochromators & Spectrographs.* URL: http://www.spectrometer.com/ [cited 27 January 2008].

[17] Honeywell International. *Honeywell Sensing and Control.* Humidity Sensors. URL: http://content.honeywell.com/sensing/prodinfo/humiditymoisture/ [cited 13 December 2007].

[18] Reiker Inc. *Electronic Inclinometer Sensors.* Reiker Incorporated Electronic Inclinometer Manufacturer. URL: http://www.riekerinc.com/Electronic Inclinometers.htm [cited 30 January 2008].

[19] Jewell Instruments. *Inclinometers.* Jewell Inclinometers: URL: http://www.jewellinstruments.com/inclinometer.htm [cited 30 January 2008].

[20] Spectron. *Inclinometers.* Spectron Glass and Electronics Incorporated. URL: http://www.spectronsensors.com/inclinometer.html [cited 30 January 2008].

[21] TSI Inc. *TSI Incorporated Innovative Measurement Solutions.* Integrating Nephelometer. URL:. http://www.tsi.com/Product.aspx?Pid=64 [cited 15 December 2007].

[22] Optec, Inc. *Optec, Inc.* Ngn-2a Nephelometer. URL: http://www.optecinc.com/optec_041.htm [cited 24 November 2007].

[23] Smith, Gregory Hallock. "Optical Designs for the Mars '03 Rover Cameras," *Current Developments in Lens Design and Optical Engineering II (Proceedings Volume)*, Vol. 4441, pp. 118-131, 2001.

[24] Fairchild Imaging. *Aerospace Applications.* URL: http://www.fairchildimaging.com/main/apps_aerospace.htm [cited 28 January 2008].

[25] Optronics. *Digital Microscope Cameras*. Medical Grade Microimaging Systems. URL: http://www.optronics.com/products/commerce.exe?search=action&category=2050&keywords=all&template=Templates/Category.html [cited 28 January 2008].

[26] Sony. *Compact Color Product Listing*. Sony Broadcast & Business Solutions Company. URL: http://bssc.sel.sony.com/BroadcastandBusiness/Display SubCategory?m=0&p=2&sp=16 [cited 30 January 2008].

[27] ASCO. *Emerson Industrial Automation. Pressure Sensors*. URL: http://www.ascovalve.com/Applications/Products/SensorsPressureSensorsData.aspx [cited 13 December 2007].

[28] Honeywell. *Honeywell Sensing and Control*. Stainless Steel Pressure Sensors Product Sheets. URL: http://sensing.honeywell.com/index.cfm?ci_id=140264&defId=121030 [cited 15 December 2007].

[29] Omron. *OMRON Electronics Components Web*. Differential Pressure Flow Sensor. URL: http://www.omron.com/ecb/products/sensor/8.html [cited 18 December 2007].

[30] GE. *GE Sensing*. Dual Analog/Digital Output Pressure Transducer. URL: http://www.gesensing.com/products/pdcr3500.htm?bc=bc_drucks [cited 14 December 2007].

[31] Climatronics. *Climatronics Corporation*. Atmospheric Pressure. URL: http://www.climatronics.com/Products/Sensors/atmospheric_pressure.php [cited 10 December 2007].

[32] Everitt, Francis. *The Gravity Probe B Experiment "Testing Einstein's Universe" Post Flight Analysis – Final Report*. Stanford, CA: Stanford University. March 2007.

[33] Galica, Gary. *Physical Sciences Incorporated*. Light Particle Detector. URL: http://www.psicorp.com/products/lpd-detector.shtml [cited 31 January 2008].

[34] Galica, Gary. *Physical Sciences Incorporated*. High Energy Charged Particle Spectrometer. URL: http://www.psicorp.com/products/lpd-spectrometer.shtml [cited 31 January 2008].

[35] Black Cat systems. *Radiation Detector and Software*. Specifications. URL: http://www.blackcatsystems.com/GM/page5.html [cited 31 January 2008].

[36] SEIC. *SE International Inc*. Handheld Analog Radiation Detector. URL: http://www.seintl.com/products/monitor_4.html [cited 31 January 2008].

[37] Pacific Northwest X-Ray Inc. *Pacific Northwest X-Ray Inc.* Advanced Survey Meter. URL: http://www.pnwx.com/Equipment/Test/SurveyMeters/ASM-990/ [cited 31 January 2008].

[38] eV Products. *eV Products.* X-Ray and Gamma Ray Detectors. URL: http://www.evproducts.com/standard_prods.html [cited 31 January 2008].

[39] Amptek. *Amptek Projects.* Spaceflight Instrumentation Summary. URL: http://www.amptek.com/instrum.html [cited 31 January 2008].

[40] Polimaster. *Polimaster Inc.* Personal Radiation Detector. URL: http://polimaster.us/products/searching/pm1401gn-gna-gnb.php [cited 31 January 2008].

[41] Sensata Technologies. *Sensor Products.* Refractive Index Sensor (RIS) Highlights. URL: http://www.sensata.com/products/sensors/spreeta-r.htm [cited 10 March 2008].

[42] FISO Technologies Inc. *Fiber Optic Sensors - Refractive Index.* Refractive index measurement, temperature sensor manufacturer. URL: http://www.fiso.com/index.php?module=CMS&id=26 [cited 10 March 2008].

[43] Omega. *OMEGA Engineering Technical Reference.* Thermocouple. URL: http://www.omega.com/prodinfo/ThermocoupleSensor.html [cited 19 December 2007].

[44] Lake Shore Cryotronics, Inc. *LakeShore Platinum RTDs.* Platinum RTDs – Technical Specifications. URL: http://www.lakeshore.com/temp/sen/prtdts.html [cited 18 February 2008].

[45] Goodrich Corporation. *Goodrich Sensor Systems.* Model 0146MD Miniature Multi-Purpose Temperature Sensor. URL: http://www.sensors.goodrich.com/literature/lit_pdfs/4025_Sensor_146MD.pdf [cited 20 February 2008].

[46] Rdf Corporation. *Aerospace & Military Products.* Data Sheet Index. URL http://www.rdfcorp.com/products/aerospace/aerospace.shtml [cited 1 March 2008]).

[47] Cohen, P. "Mystery of Mars rover's 'carwash' rolls on," *New Scientist*, Vol. 184, Issue 2479/2480, pp. 7, 2004.

[48] NASA. *Mars Exploration Rover Mission.* Spirit Updates. URL: http://marsrover.nasa.gov/mission/status_spiritAll.html [cited 2 December 2007].

[49] Lakdawalla, E. *The Planetary Society.* The 2007 Martian Dust Storm: Crisis for Some, Opportunity for Others. URL: http://www.planetary.org/news/2007/0720_The_2007_Martian_Dust_Storm_Crisis_for.html [cited 2 December 2007].

[50] Omega. *OMEGA Selection Guide.* Portable Velocity Meters. URL: http://www.omega.com/guides/portablevelocitymeters_compareall.html [cited 21 Decmber 2007].

[51] Calright. *Calright Instruments.* Anemometer/Airflow. URL: http://www.calright.com/_coreModules/common/categoryDetail.aspx?entityType=6&categoryID=39 [cited 11 Decmber 2007].

[52] Symmetricom. *Symmetricom Timing, Test and Measurement Division.* Precision Frequency References. URL: http://www.symmttm.com/products_precision_frequency_references.asp [cited 23 February 2008].

[53] Symmetricom. *Symmetricom.* Space, Defense and Avionics. URL: http://www.symmsda.com/applications/space.asp [cited 23 February 2008].

[54] Analog Devices. *Analog Devices.* 12.5 MIPS DSP Microprocessor. URL: http://www.ortodoxism.ro/datasheets/analogdevices/ADSP-2100KG.pdf [cited 4 March 2008].

[55] Xepoc Technology Inc. *ChipDocs Datasheets for Electronic Components.* ADSP-2100 Series Datasheets. URL: http://www.chipdocs.com/datasheets/datasheet-pdf/Analog-Devices/ADSP-2100.html [cited 5 March 2008].

[56] Ball, Andrew J., Garry, James R., et al. *Planetary Landers and Entry Probes.* New York, New York: Cambridge University Press, 2007.

[57] European Space Agency. *Cassini-Huygens.* Instrument Introduction. URL: http://sci.esa.int/science-e/www/object/index.cfm?fobjectid=31193 [cited 15 March 2008].

[58] NASA. *Cassini-Huygens Mission to Saturn and Titan.* Spacecraft – Huygens Probe Instruments. URL: http://saturn.jpl.nasa.gov/spacecraft/instruments-huygens.cfm [cited 10 March 2008].

[60] Fairchild Imaging. *Fairchild Imaging Press Release.* Fairchild Imaging CCD on Cassini Captures Images of Titan in Close Flyby. URL: http://www.fairchildimaging.com/news/2005/apr/19titanflyby.htm [cited 20 January 2008].

[61] Fairchild Imaging. *Fairchild Imaging Products.* CCD 424. URL: http://www.fairchildimaging.com/products/fpa/custom/ccd_424.htm [cited 20 January 2008].

[62] Bird, M.K., Dutta-Roy, R., et al. "The Huygens Doppler Wind Experiment: Titan Winds Derived from Probe Radio Frequency Measurements," *Space Science Reviews*, Vol. 104, pp. 613-640, 2001.

[63] DWE Website. *The Doppler Wind Experiment DWE on CASSINI/ HUYGENS.* Transmitter and Receiver USO programs. URL: http://www.astro.uni-bonn.de/%7Edwe/dweid/node14.html#SECTION00041000000000000000 [cited 4 March 2008].

[64] Haberman, John. Atmospheric Experiments Laboratory NASA Goddard Space Flight Center. Gas Chromatograph Mass Spectrometer (GCMS). URL: http://aeb.gsfc.nasa.gov/saturnGCMS.shtml [cited 24 February 2008].

[65] Niemann, H B., Atreya, S K., et al. (2002). "The Gas Chromatograph Mass Spectrometer for the Huygens Probe," *Space Science Reviews*, Vol. 104. pp. 553-591, 2001.

[66] Harpold, Dan. "Cassini-Huygens Gas Chromatograph Mass Spectrometer," *Presentation from 5th Workshop on Harsh-Environment Mass Spectrometry*, Lido Beach Sarasota, FL, HEMS, 2005.

[67] European Space Agency. *Cassini-Huygens.* Instrument Gas Chromatograph Mass Spectrometer. URL: http://sci.esa.int/science-e/www/object/index.cfm?fobjectid=31193&fbodylongid=736 [cited 15 March 2008].

[68] Niemann, H., Demick, J., et al. "The Cassini-Huygens Probe Gas Chromatograph Mass Spectrometer (GCMS) Experiment, First Results," *Abstract for the 36th Lunar and Planetary Science Conference.* League City, Texas, LPSC, 2005.

[69] Fulchignoni, M., Ferri, F. et al. "The Characterization of Titan's Atmospheric Physical Properties by the Huygens Atmospheric Structure Instrument (HASI)," *Space Science Reviews*, Vol. 104, pp. 395-431, 2005.

[70] Fulchignoni, M., Angrilli, F., et al. "The Huygens Atmospheric Structure Instrument (HASI)," In: J. Lebreton (ed.), Huygens, Science, Payload, and Mission, (pp. 163-176), Noordwijk, The Netherlands: ESA Publications Division ESTEC, 1997.

[71] Zarnecki, J.C., Ferri, F., et al. "In-Flight Performance of the HASI Servo Accelerometer and Implication for Results at Titan," *Proceedings from International Workshop Planetary Probe Atmospheric Entry and Descent Trajectory Analysis and Science.* Lisbon, Portugal, IPPW, 2003.

[72] Honeywell. *Honeywell Thermal Switch Products Aerospace – Redmond (Defense & Space).* The Evolution of the Thermal Switch from United Control to Honeywell. URL: http://www.thermalswitch.com/tshistory.shtml [cited 17 March 2008].

[73] Endevco Corporation. *Endevco*. 7264-2000. URL: http://www.endevco.com/product/Product.aspx?product_id=71 [cited 15 March 2008].

[74] Angrilli, F., Aboudan, A., et al. "HASI Engineering Operations at Titan," *Presentation from Space Ops 2006 Conference*, AIAA, 2006.

[75] Goodrich Corporation. *Goodrich*. Sensor Systems. URL: http://www.sensors.goodrich.com/ [cited 1 March 2008].

[76] Lopez-Moreno, J.J., Molina-Cuberos, G.J., et al., "The Comas-Sola Mission to Test the Huygens/HASI Instrument on Board a Stratospheric Balloon". URL: http://saturn.iwf.oeaw.ac.at/iwfmag/cassini/paper1.doc [cited 5 March 2008].

[77] European Space Agency. *Cassini-Huygens*. Instrument Huygens Atmosphere Structure Instrument. URL: http://sci.esa.int/science-e/www/object/index.cfm?fobjectid=31193&fbodylongid=737 [cited 15 March 2008].

[78] Hofe, Robin. "Signal Analysis of the Electric and Acoustic Field Measurements by the Huygens Instrument HASI/PWA," Master's Thesis. Institute of Broadband Communication University of Technology, Graz, Austria, 2005. (Unpublished).

[79] European Space Agency. *Cassini-Huygens*. Instruments SSP: Surface Science Package. URL: http://sci.esa.int/science-e/www/object/index.cfm?fobjectid=31193&fbodylongid=740 [cited 15 March 2008].

[80] The Open University. Planetary and Space Sciences Research Institute. "Huygens-SSP Experimenter to Archive Interface Control Document Draft 1," URL: http://pds.nasa.gov/documents/pag/sspeaicd.doc [cited 16 March 2008].

[81] Zarnecki, J.C., Banaszkiewicz, M., et al. "The Huygens Surface Science Package". In: J. Lebreton (ed.), Huygens, Science, Payload, and Mission. (pp. 177-195), Noordwijk, The Netherlands: ESA Publications Division ESTEC. 1997.

[82] Spectron. (2008). Mission Accomplished. Spectron shares in success as Huygens Probe makes triumphant landing on Titan. URL: http://www.spectronsensors.com/articles/Cassini-Huygens%20Lands.pdf [cited 13 March 2008].

[83] Ringrose, T.J., Hathi, B., et al., "Thermal Properties Instrument for Measurement on Titan," *Abstract for the 31st Lunar and Planetary Science Conference*. Houston, Texas, LPSC, 2000.

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

International Workshop Planetary Probe Atmospheric Entry and Descent Trajectory Analysis and Science, Lisbon, Portugal, IPPW, 2003.

[85] Butlin, Chris, *The Cassini-Huygens Mission to Saturn and its Moons.* Determining the content of Titan's oceans with a Refractometer - Teachers' Guide. URL: http://www.bnsc.gov.uk/assets/channels/education/Huygens_Refractometer_Teachers_guide.doc [cited 17 March 2008].

[86] Hamamatsu. *NMOS Linear Image Sensor.* S3921/S3924 Series Datasheet. URL: http://sales.hamamatsu.com/assets/pdf/parts_S/S3924_series.pdf [cited 17 March 2008].

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 out puts 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 in shown in Figure A-2.

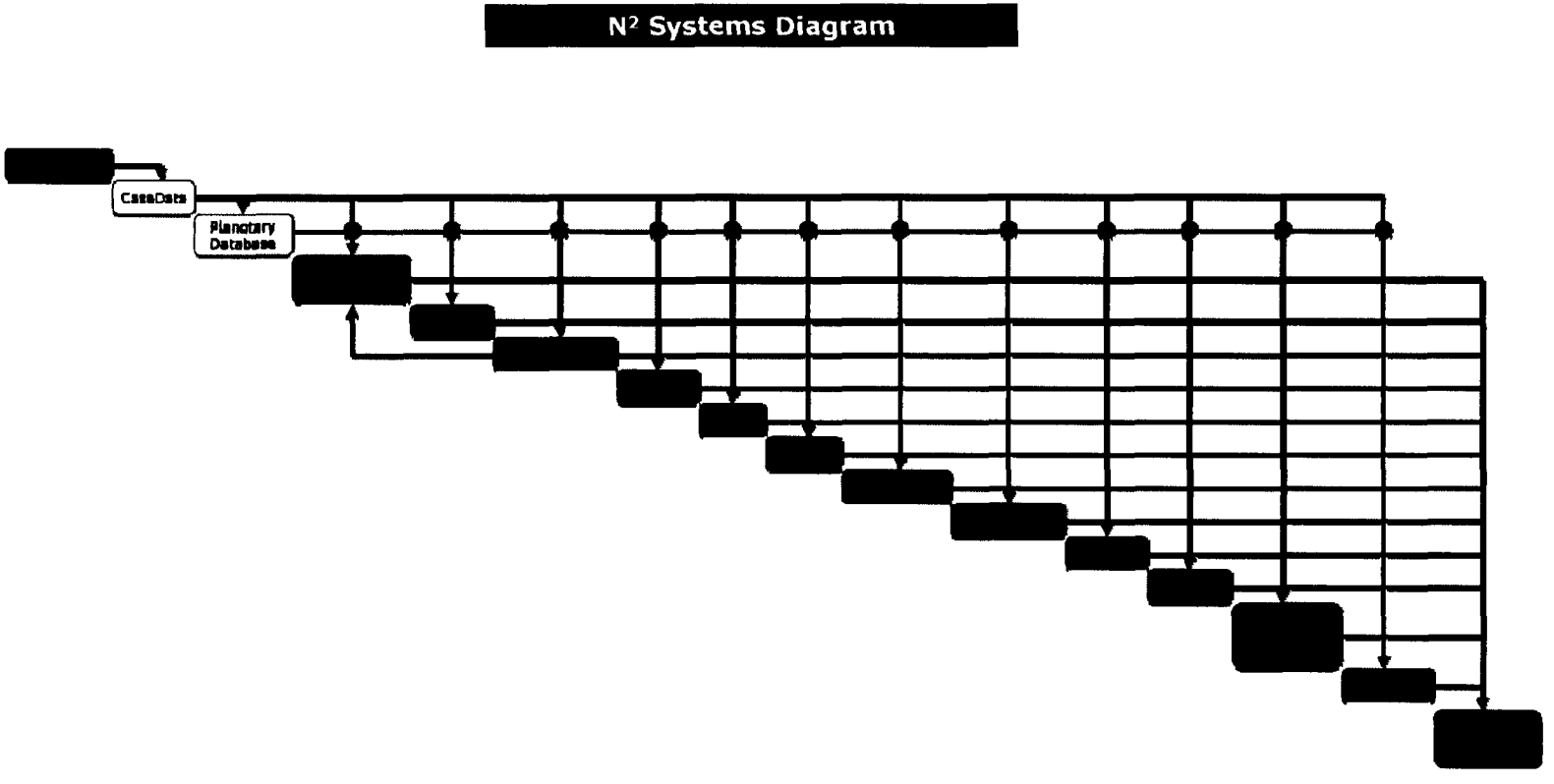


Figure A-1 ISSPO Systems N2 Flowchart

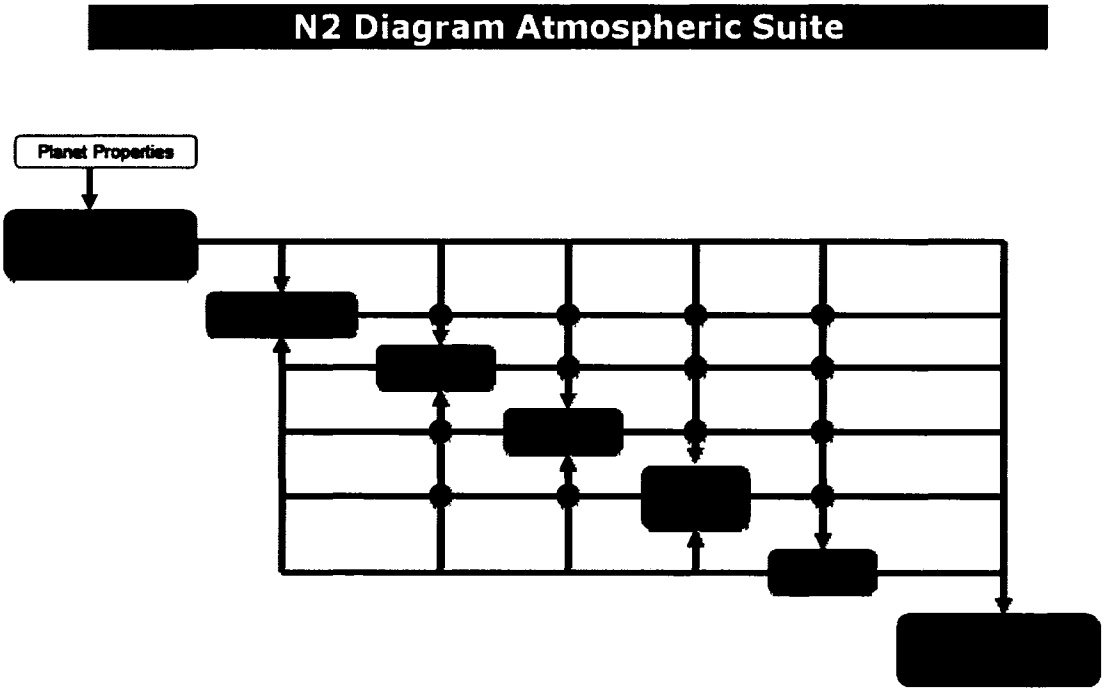


Figure A-2 Atmospheric Suite N^2 Diagram

Circular nodes in the two diagrams indicate a connection between data flow between the different sensor modules. These sensors are often design 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 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';
UNITS = 'SI';                  %  'SI' or 'British'  Units
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'    'ARRAY'        'LOW'        'VISUAL'     'UV'    'NIR'   };
```

Output File:

```
   ********************************************************************************
   ***   I N - S I T U   S E N S O R   P A Y L O A D   O P T I M I Z A T I O N   T O O L   ***
   ********************************************************************************

***   P R O G R A M   I N P U T S   ***
  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

*** P A Y L O A D S E N S O R B R E A K D O W N ***

SENSOR	MASS	VOLUME	POWER
	kg	m^3	W
OPTICS	0.0000	0.0000	0.0000
SENSOR PAYLOAD TOTAL	0.0000	0.0000	0.0000

*** 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.

*** P L A N E T A R Y D A T A B A S E S U M M A R Y R E S U L T S ***

*** I N P U T S ***
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^2
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 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 :	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 PPROPERTIES ***

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

Element	% Composition
N2	98.400 %
CH4	1.600 %

Trace Elements

Element	Amount	Concentration

*** 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:

```
   ********************************************************************************************
   ***   I N - S I T U   S E N S O R   P A Y L O A D   O P T I M I Z A T I O N   T O O L   ***
   ********************************************************************************************

***   P R O G R A M   I N P U T S   ***
  User Input Filename:        Huygens_DWE
  Planetary Body Name:        Titan
  Unit System:                SI
  Print Summary Option:       N

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

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


   SENSOR PACKAGES DESIGNED:
-------------------------------------------------------------------------
     ATMOSPHERE
```

- Temperature
- Pressure
- Wind Velocity
- Humidity
- Density

*** P A Y L O A D S E N S O R B R E A K D O W N ***

SENSOR	MASS	VOLUME	POWER
	kg	m^3	W
ATMOSPHERE			
- Temperature	0.0000	0.0000	0.0000
- Pressure	0.0000	0.0000	0.0250
- Wind Velocity Probe	1.8000	0.0006	35.0000
- Humidity	0.0000	0.0000	0.0000
- Density	5.0000	0.0070	0.8250
SENSOR PAYLOAD TOTAL	5.9000	0.0076	35.8500

*** CASE LIMITS SUMMARY ***

****

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 in Design.

CASE VOLUME LIMIT: FAILED!

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.

CASE POWER LIMIT: FAILED!

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.

*** P L A N E T A R Y D A T A B A S E S U M M A R Y R E S U L T S ***

*** I N P U T S ***
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^2
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 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 :	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 PPROPERTIES ***

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

Element	% Composition
N2	98.400 %
CH4	1.600 %

Trace Elements

Element	Amount	Concentration

*** W I N D V E L O C I T Y 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 ***
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:	7.280	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:

```
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:

```
   ********************************************************************************************
   ***   I N - S I T U   S E N S O R   P A Y L O A D   O P T I M I Z A T I O N   T O O L   ***
   ********************************************************************************************

***   P R O G R A M   I N P U T S   ***
  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

*** P A Y L O A D S E N S O R B R E A K D O W N ***

SENSOR	MASS	VOLUME	POWER
	kg	m^3	W
GAS ANALYSIS	17.3000	0.0184	71.0000
SENSOR PAYLOAD TOTAL	17.3000	0.0184	71.0000

*** 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.

*** P L A N E T A R Y D A T A B A S E S U M M A R Y R E S U L T S ***

*** I N P U T S ***
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^2
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 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 :                   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 PPROPERTIES  ***
  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
------------------------------
  Element   % Composition
      N2          98.400 %
     CH4           1.600 %

 Trace Elements
------------------------------
  Element      Amount     Concentration



   ***  G C M S  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  ***
  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
```

High:	141.00	amu	
Mass Filter Type:	Quadrupole		
Detector Operational Type:	Electron Multiplier		
Dynamic Mass-Charge Scanning Range:	8	N/A	
Scanning Resolution:	0.00	amu	
Scanning Sensitivity:	0.00	A/Torr	
Minimum Detected Partial Pressure:	0.00	Torr	
Operating Pressure:	0.00	Torr	
Maximum Operating Temperature:	100.00	deg C	
Number of Ion Sources:	5	N/A	
Charge Field Range - Low:	1.00	W Degas	
High:	16.00	W Degas	
Electron Source Voltage - Low:	25.00	V	
High:	70.00	V	
Ion Energy:	8.00	12.00	V
Electron Source Current - Low:	0.00	mA	
High:	0.08	mA	
Warm Up Period Mass Stability:	0.00	amu	
Time:	36.00	min	
Sensor Return Data Rate:	960.00	bits/sec	

Physical Properties		
Sensor Mass:	17.30	kg
Sensor Dimensions - Length:	47.00	cm
Width:	19.80	cm
Height:	19.80	cm
Power Source Voltage:	28.00	V
Power Source Voltage Type:		DC
Power Source Current:	1.68	A
Power Requirements - Typical:	28.00	W
Average:	41.00	W
Peak:	71.00	W

Sensor Comments:
Combined Gas Chromatograph Mass Spectrometer
Probe design consists of an 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

4. Huygens Atmosphere Structure Instrument (HASI)

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:

```
function Huygens_HASI

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

MASS_LIMIT = 6.3;            %  Max Allowable Sensor Package Wt 'lbm' or 'kg'
POWER_LIMIT = 85;            %  Max Total Power Available 'BTU's/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'       ;
                  'EM FIELD'         'N/A'             'N/A'              ;
                  'ACCELERATION'     'MEDIUM'          'Const Current'    ;
                  'ACOUSTICS'        'SENSOR'          'N/A'              ;
                  'DATA PROCESSING'  'LOW'             4                  };
```

Output File:

```
   ********************************************************************************************
   ***   I N - S I T U   S E N S O R   P A Y L O A D   O P T I M I Z A T I O N   T O O L   ***
   ********************************************************************************************

***   P R O G R A M   I N P U T S   ***
  User Input Filename:        Huygens_HASI
  Planetary Body Name:        Titan
  Unit System:                SI
  Print Summary Option:       N

  Payload Mass Limit:           6.300   kg
  Payload Volume Limit:        0.5000   m^3
  Payload Power Limit:         85.000   W

*************************************************************************
```

SENSOR PACKAGES DESIGNED:

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

ACCELERATION
ACOUSTICS
DATA PROCESSING

*** P A Y L O A D S E N S O R B R E A K D O W N ***

SENSOR	MASS	VOLUME	POWER
	kg	m^3	W
ATMOSPHERE			
- Temperature	0.0004	0.0000	0.0600
- Pressure	0.0000	0.0000	0.0250
- Wind Velocity Probe	0.2270	0.0004	0.0000
Handheld Unit	0.0000	0.0003	0.0000
- Humidity	0.0000	0.0000	0.0000
- Density	5.0000	0.0070	0.8250
ACCELERATION	0.0710	0.0000	0.4800
ACOUSTICS	0.0040	0.0000	0.0000
DATA PROCESSING	0.0000	0.0000	0.7900
SENSOR PAYLOAD TOTAL	5.3023	0.0077	2.1800

*** 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.

*** P L A N E T A R Y D A T A B A S E S U M M A R Y R E S U L T S ***

*** I N P U T S ***
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^2
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 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 :	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 PPROPERTIES ***

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

Element	% Composition
N2	98.400 %
CH4	1.600 %

Trace Elements

Element	Amount	Concentration

*** T E M P E R A T U R E 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 ***
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:
Resistance type Thermal Sensor. High Reliability, Linear Output, Miniature Size, Wide Temperature Range, Recommended for Space Vehicles.

*** P R E S S U R E 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   ***
  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:      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

  Sensor Comments:
  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.
```

*** W I N D V E L O C I T Y 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 ***
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.

*** H U M I D I T Y 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 ***
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
```



```
   ***  D E N S I T Y   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  ***
  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:             0.0020000  g/cc
  Sensor Tolerance:                 0.000600   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

*** A C C E L E R O M E T E R 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 ***

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	Type
Current Range - Low:	16.00	mA
High:	16.00	mA
Bias Voltage:	28.00	V
Bias Voltage Type:	DC	Type
Electrical Noise:	0.30	mG's
Electrical Power:	0.48	W
Environmental Properties		
Sensor Temperature Range - Low:	-55.00	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:
Single Axis, Piezoresistive accelerometer. Long-term repeatability and superior reliability. Dual built-in self-test. Flight Proven - Huygens Lander Descent phase HASI.

*** A C O U S T I C 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 ***
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
	Tolerance - 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:	34.300	mm
	Width:	9.500	mm

Height:	9.500	mm
Sensor Mass:	4.000	gram

Sensor Comments:
Pressure Sensor. Excellent for extreme conditions - temperature, pressure. Not suited for quiet sounds. Flight Proven - Huygens Lander.

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

*** I N P U T S ***

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:

```
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 'ft^3' or 'm^3'


%  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:

```
   ********************************************************************************************
   ***   I N - S I T U   S E N S O R   P A Y L O A D   O P T I M I Z A T I O N   T O O L   ***
   ********************************************************************************************

***   P R O G R A M   I N P U T S   ***
  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   m^3
  Payload Power Limit:         11.000   W

*************************************************************************
```

SENSOR PACKAGES DESIGNED:

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

INCLINATION
ACCELERATION
ACOUSTICS
REFRACTION

*** P A Y L O A D S E N S O R B R E A K D O W N ***

SENSOR	MASS	VOLUME	POWER
	kg	m^3	W
ATMOSPHERE			
- Temperature	0.0004	0.0000	0.0600
- Pressure	0.0000	0.0000	0.0250
- Wind Velocity Probe	0.2270	0.0004	0.0000
Handheld Unit	0.0000	0.0003	0.0000
- Humidity	0.0000	0.0000	0.0000
- Density	5.0000	0.0070	0.8250
INCLINATION	0.0000	0.0000	0.0000
ACCELERATION	0.0270	0.0000	0.0000
ACOUSTICS	0.0320	0.0000	0.0000
REFRACTION	0.0000	0.0012	10.0000
SENSOR PAYLOAD TOTAL	5.2864	0.0089	10.9100

*** CASE LIMITS SUMMARY ***

****

```
   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 in
Design.
   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.

   ***  P L A N E T A R Y   D A T A B A S E   S U M M A R Y   R E S U L T S  ***

***  I N P U T S  ***
  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^2
  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 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 :                   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 PPROPERTIES  ***
  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
-----------------------------
  Element    % Composition
     N2          98.400 %
    CH4           1.600 %

 Trace Elements
-----------------------------
  Element       Amount      Concentration



   ***   T E M P E R A T U R E   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   ***
  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:
```

Resistance type Thermal Sensor. High Reliability, Linear Output, Miniature Size, Wide Temperature Range, Recommended for Space Vehicles.

*** P R E S S U R E 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 ***

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:	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

Sensor Comments:
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.

*** W I N D V E L O C I T Y 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 ***
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.

*** H U M I D I T Y 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 ***
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



   ***   D E N S I T Y   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   ***
  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:           0.0020000  g/cc
  Sensor Tolerance:               0.000600   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

*** I N C L I N O M E T E R 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 ***

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
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

*** A C C E L E R O M E T E R 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 ***
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	Type
Current Range - Low:		0.00	mA
	High:	0.00	mA
Bias Voltage:		0.00	V
Bias Voltage Type:		Charge	Type
Electrical Noise:		0.30	mG's
Electrical Power:		0.00	W

Environmental Properties

Sensor Temperature Range - Low:		-269.00	deg C
	High:	260.00	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.

*** A C O U S T I C 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 ***

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

	Height:	21.400	mm
Sensor Mass:		32.000	gram

Sensor Comments:
General Purpose Acoustic Sensor. High Frequency sensor. Robust, Reliable Sensor.

*** R E F R A C T I O N 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 ***

Unit System: SI

****

** Sensor Properties **

Refraction Sensor Type:	REF-Huy	
Sensing Refraction Range - Min:	1.25000	N/A
Middle:	1.45000	N/A
Sensing Resolution:	1.0000E-003	RIU
Sensing Drift Rate:	0.00000	RIU
Sensor Accuracy:	1.00000	%

Power Requirements

Input Voltage:	15.000	V
Voltage Type:	DC	Type
Input Current:	6.660	mA
Input Power:	10.000	mW

Physical Properties

Sensor Mass:	0.00	g
Sensor Dimensions - Length:	10.00	cm
Width:	10.00	cm
Height:	11.61	cm

Environmental Properties

Temperature Range - Low:	-40.00	deg C
High:	65.00	deg C

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:

```
function Venus_ATM

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

MASS_LIMIT = 30;            %  Max Allowable Sensor Package Wt 'lbm' or 'kg'
POWER_LIMIT = 200;           %  Max Total Power Available 'BTU's/hr' or 'W'
VOLUME_LIMIT = 0.50;            %  Max Allowable Volume 'ft^3' or 'm^3'


%  Type of Science Data to be returned
SENSOR_DATA = {   'ATMOSPHERE'     'WIND VELOCITY'     'ANEMOMETER'     'TEMPERATURE'     'VOLTAGE'      'N/A'          ;
                  'INCLINATION'    'MEDIUM'            'N/A'            'N/A'             'N/A'          'N/A'          ;
                  'ACCELERATION'   'MEDIUM'            'Voltage'        'N/A'             'N/A'          'N/A'          ;
                  'EM FIELD'       'N/A'               'N/A'            'N/A'             'N/A'          'N/A'          ;
                  'RADIATION'      'Charged Particle'  'N/A'            'N/A'             'N/A'          'N/A'          ;
                  'OPTICS'         'ARRAY'             'MEDIUM'         'VISUAL'          'UV'           'NIR'          };
```

Output File:

```
   ********************************************************************************************
   ***   I N - S I T U   S E N S O R   P A Y L O A D   O P T I M I Z A T I O N   T O O L   ***
   ********************************************************************************************

***   P R O G R A M   I N P U T S   ***
  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

*** P A Y L O A D S E N S O R B R E A K D O W N ***

SENSOR	MASS	VOLUME	POWER
	kg	m^3	W
ATMOSPHERE			
- Temperature	0.0000	0.0000	0.0000
- Pressure	0.0000	0.0000	0.0300
- Wind Velocity Probe	0.5000	0.0004	0.1000
Handheld Unit	0.0000	0.0000	0.0000
- Humidity	0.0000	0.0000	0.0000
- Density	5.0000	0.0070	0.8250
INCLINATION	0.0000	0.0000	0.0000
ACCELERATION	0.1417	0.0000	0.6400
EM FIELD	0.3100	0.0004	0.0000
Handheld Unit	0.7700	0.0012	0.0000

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.

*** P L A N E T A R Y D A T A B A S E S U M M A R Y R E S U L T S ***

*** I N P U T S ***
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+006	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 : 0

*** PLANETARY ORBITAL PARAMETERS ***

Semi-Major Axis :	1.08210e+011	m
Sidereal Orbit Period :	2.24701e+002	Earth Days
Tropical Orbit Period :	2.24695e+002	Earth Days
Perihelion Distance :	1.07480e+011	m
Aphelion Distance :	1.08940e+011	m
Synodic Period :	5.83920e+002	Earth Days
Mean Orbital Velocity :	3.50200e+004	m/s
Maximum Orbital Velocity :	3.52600e+004	m/s
Minimum Orbital Velocity :	3.47900e+004	m/s
Orbital Inclination :	3.39000e+000	deg
Orbital Eccentricity :	6.70000e-002	
Sidereal Rotation Period :	-5.83250e+003	Earth Hours
Length of Day :	2.80200e+003	Earth Hours
Obliquity to Orbit :	1.77360e+002	deg

*** ATMOSPHERIC PPROPERTIES ***

Planetary Surface Temperature :	464.00	deg C
Planetary Surface Pressure :	9.20000e+006	Pa
Planetary Surface Density	6.44000e+001	kg/m^2

ATMOSPHERIC COMPOSITION

Major Elements

Element	% Composition
CO2	96.500 %
N2	3.500 %

Trace Elements

Element	Amount	Concentration
SO2	150.000	ppm
Ar	70.000	ppm
H2O	20.000	ppm
CO	17.000	ppm
He	12.000	ppm
Ne	7.000	ppm

*** T E M P E R A T U R E 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 ***

Planetary Surface Temperature: 464.00 deg C

Unit System: SI

****

```
** Sensor Properties **
  Thermal Sensor Type:     K
  Thermal Sensor Low Limit:    -200.000   deg C
  Thermal Sensor High Limit:   1250.000   deg C
  Power Requirements:             0.000   W
  Mass Requirements:              0.000   g
  Dimensions - Length:            0.000   mm
               Width:             0.000   mm
              Height:             0.000   mm
  Operating Pressure:             0.000   Pascals
  Error Tolerance:                0.750   %
                                  2.200   deg C
  TRL Level:                          0     TRL

Thermocouple Construction Materials
  Positive Terminal:              Ni-Cr    +Lead
  Negative Terminal:              Ni-Al    -Lead

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
  Insulation Type:                GG
  Insulation Lower Limit:          -73.000   deg C
  Insulation Upper Limit:          482.000   deg C

  Environmental Properties
  Abrasion Resistance:                Poor
  Flexibility:                        Good
  Water Submersion:                   Poor
  Resistance to Solvents:             Excellent
  Resistance to Acids:                Excellent
  Resistance to Bases:                Excellent
  Resistance to Flame:                Excellent
  Resistance to Humidity:             Fair
  Overall Material Construction:      Glass Braid
  Conductor Material Construction:    Glass Braid



   ***   P R E S S U R E   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   ***
  Planetary Surface Pressure:   9200000.00  Pa
  Unit System:          SI

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


** Sensor Properties **
  Pressure Sensor Type:                13(C,U)2000P
```

Pressure Sensor Accuracy:	2.000	%
Pressure Sensor Stability:	0.600	% @ 1 yr
Over Pressure Rated Value:	3.000	N/A
Burst Pressure Rating:	5.000	N/A
Life Cycles:	1.000E+006	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:
Low cost. Rugged Isolated Stainless Steel Design. Calibrated and temperature compensated. Oil-isolated housing. Measurement in hostile environments.

*** W I N D V E L O C I T Y 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 ***

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:	0	N/A
Battery Type:	N/A	Type
Voltage Input:	7.000	V
Voltage Type:	DC	N/A
Current Input:	14.000	mA
Input Power:	100.000	mW

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

*** H U M I D I T Y 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 ***
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

*** D E N S I T Y 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 ***

Density Sensor Operating Type:	GAS	
Planetary Surface Density:	64.40	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:	0.0020000	g/cc
Sensor Tolerance:	0.000600	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

*** I N C L I N O M E T E R 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 ***
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

*** A C C E L E R O M E T E R 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 ***

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

Power Requirements

Input Type:		Voltage	Type
Current Range -	Low:	4.00	mA
	High:	20.00	mA
Bias Voltage:		32.00	V
Bias Voltage Type:		DC	Type
Electrical Noise:		0.30	mG's
Electrical Power:		0.64	W

Environmental Properties

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

Physical Properties

Sensor Mass:		141.700	gr
Sensor Dimensions -	Length:	5.690	cm
	Width:	2.896	cm
	Height:	2.896	cm

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

*** A C F I E L D 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 ***

Magnetic Field Strength:	0.000	mGauss

Unit System: SI

****

** 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:	16.380	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-STD-975). Fully Space Qualified

*** R A D I A T I O N 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 ***
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:                1   # 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     0.100     0.300    cm^2 sr

  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
```

*** 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:           MEDIUM
  Image Type:           VISUAL   UV   NIR
  Unit System:          SI

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

** Sensor Properties **
  Imaging Sensor Type:              CCD 3041
```

Imaging Spectrum:	X-RAY	UV	VISUAL	NIR	MICRO
Array Dimensions - Length:	2048	# Pixels			
Width:	2048	# Pixels			
Sensor Pixel Size:	15.0000	micro-m			
Imaging Array Size - Length:	30.720	mm			
Width:	30.720	mm			
Read Noise at 1 MHz:	7	e-			
250 kHz:	0	e-			
Full Well Capacity - Pixel:	0	ke-			
Register:	750	ke-			
Sensor Gain:	0.00	e-/ADU			
Sensor Linearity:	0.00	%			
ADC Dynamic Range:	N/A	N/A			
Readout Rates:	2.0000	1.0000	MHz		
Readout Time:	1.100000	0.364000	0.180000	sec	
Frame Rates:	0.90	2.70	5.60	fps	

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 - Length:	45.72	mm
Width:	36.83	mm
Height:	2.01	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:
- 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
- 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('*************************************************************************\n');
fprintf('**    IIIIIIII     SSSSSS      SSSSSS     PPPPPPP      OOOOO      **\n');
fprintf('**       II       SS    SS    SS    SS    PP    PP    OO   OO     **\n');
fprintf('**       II       SS          SS          PP    PP   OO     OO    **\n');
fprintf('**       II        SSSSSS      SSSSSS     PPPPPPP    OO     OO    **\n');
fprintf('**       II             SS          SS    PP         OO     OO    **\n');
fprintf('**       II       SS    SS    SS    SS    PP          OO   OO     **\n');
fprintf('**    IIIIIIII     SSSSSS      SSSSSS     PP           OOOOO      **\n');
fprintf('*************************************************************************\n');


fprintf('***   IN - SITU SENSOR PAYLOAD OPTIMIZATION TOOL   ***');

InputFile = input('\n\nEnter a Input Filename to use: ', 's');
```

```
if exist(InputFile) == 2
    fprintf('\n\n***Loading Input Design Parameters from File.***\n');
    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 == 18
    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_PRG(200);
        return;
    end
else
    ERROR_PRG(100);
    return;
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') == 1
        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('HUMIDITY_SENSOR_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('DENSITY_SENSOR_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},'ACOUSTICS') == 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');
       %  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
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');
       %  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
end
```

```
    %  Runs the Acceleration Sensor Program
    if strcmpi(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;
       end
          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};
          %  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
```

```
          else
        ERROR_PRG(109);
        return;
          end
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
%    cntr = 0;
    cntr = 1;
    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
        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
    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_PRG(113);
            return;
               end
    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{i,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');
            %  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
               else
            ERROR_PRG(114);
            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
MASS_PROPERTIES(SENSOR_DATA,UNITS);
if exist('MASS_PROPERTIES_Data.mat') == 2
    load('MASS_PROPERTIES_Data.mat');
else
    ERROR_PRG(215);
    return;
end


%   Determine Each Sensors Power Requirements
POWER_PROPERTIES(SENSOR_DATA,UNITS);
if exist('POWER_PROPERTIES_Data.mat') == 2
    load('POWER_PROPERTIES_Data.mat');
else
```

```
    ERROR_PRG(216);
    return;
end


%  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!'    ;
                     'Minimum Sensor Requirements met allows for System Redundancy in Components.'};
else
    MASS_FLAG = 'FAILED!';
    MASS_FLAG_STR = {'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 Design.'};
end

if TOTAL_VOLUME <= VOLUME_LIMIT
    VOLUME_FLAG = 'SUCCEEDED!';
    VOLUME_FLAG_STR = {'Designed Mission Sensor Package Meets Volume Limit!'    ;
                       '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.'};
end

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

%*************************************************************************************************************************
%*********        PRINT SUMMARY SECTION TO SCREEN


fprintf('\n\n\n   ********************************************************************************************');
fprintf('\n   ***   I N - S I T U   S E N S O R   P A Y L O A D   O P T I M I Z A T I O N   T O O L   ***');
fprintf('\n   ********************************************************************************************');
fprintf('\n\n***   P R O G R A M   I N P U T S   ***');
fprintf('\n  User Input Filename:        %s', InputFile');
fprintf('\n  Planetary Body Name:        %s', BODY);
fprintf('\n  Unit System:                %s', UNITS);
fprintf('\n  Print Summary Option:       %s', PRINTFLG);

fprintf('\n\n  Payload Mass Limit:      %10.3f   %s', MASS_LIMIT, MassUnit);
fprintf('\n  Payload Volume Limit:    %10.4f   %s', VOLUME_LIMIT, VolUnit);
fprintf('\n  Payload Power Limit:     %10.3f   %s', POWER_LIMIT, PowerUnit);

fprintf('\n\n*************************************************************************\n\n');


fprintf('\n   SENSOR PACKAGES DESIGNED:');
fprintf('\n-------------------------------------------------------------------------');

for i = 1:size(SENSOR_DATA,1)
    fprintf('\n     %s', SENSOR_DATA{i,1});
    if strcmpi(SENSOR_DATA{i},'ATMOSPHERE') == 1
        fprintf('\n        - Temperature');
        fprintf('\n        - Pressure');
        fprintf('\n        - Wind Velocity');
        fprintf('\n        - Humidity');
        fprintf('\n        - Density');
    end
    if strcmpi(SENSOR_DATA{i},'OPTICS') == 1
        fprintf('\n        - %s', SENSOR_DATA{i,2});
        for r = 3:size(SENSOR_DATA,2)
            fprintf('\n        - %s', SENSOR_DATA{i,r});
        end
```

```
    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('\n        - %s', SENSOR_DATA{i,r});
            else
                break;
            end
        end
    end
end


%   Print  Mass Volume Breakdown Summary
k = 1;
fprintf('\n\n\n                          ***   P A Y L O A D   S E N S O R   B R E A K D O W N   ***');
fprintf('\n\n|=================================|============================|============================|==================
==========|');
fprintf('\n|             SENSOR              |            MASS            |            VOLUME          |            POWER
|');
fprintf('\n|=================================|============================|============================|==================
=========|');
fprintf('\n|---------------------------------|          %4s              |          %6s              |          %6s
|', MassUnit,VolUnit,PowerUnit);
fprintf('\n|=================================|============================|============================|==================
=========|');
for i = 1:size(SENSOR_DATA,1)
    if strcmpi(SENSOR_DATA{i,1},'ATMOSPHERE') == 1
        fprintf('\n|                                 |                            |                            |
|');
        fprintf('\n|     %17s           |                            |                            |
|',SENSOR_DATA{k,1} );
        fprintf('\n|         - Temperature           |        %10.4f        |        %10.4f        |        %10.4f
|',MASS_PROPS(k,1),VOLUME_SUM(k),POWER_PROPS(k));
        fprintf('\n|         - Pressure              |        %10.4f        |        %10.4f        |        %10.4f
|',MASS_PROPS(k+1,1),VOLUME_SUM(k+1),POWER_PROPS(k+1));
        if strcmpi(WV_SENS_TYPE,'ANEMOMETER') == 1
            fprintf('\n|         - Wind Velocity Probe  |        %10.4f        |        %10.4f        |        %10.4f
|',MASS_PROPS(k+2,1),VOLUME_SUM(k+2),POWER_PROPS(k+2));
            fprintf('\n|             Handheld Unit       |        %10.4f        |        %10.4f        |        %10.4f
|',MASS_PROPS2(1),VOLUME_SUM2,POWER_PROPS2);
        end
        if strcmpi(WV_SENS_TYPE,'DOPPLER') == 1
            fprintf('\n|         - Wind Velocity Probe  |        %10.4f        |        %10.4f        |        %10.4f
|',MASS_PROPS(k+2,1),VOLUME_SUM(k+2),POWER_PROPS(k+2));
        end
        fprintf('\n|         - Humidity              |        %10.4f        |        %10.4f        |        %10.4f
|',MASS_PROPS(k+3,1),VOLUME_SUM(k+3),POWER_PROPS(k+3));
        fprintf('\n|         - Density               |        %10.4f        |        %10.4f        |        %10.4f
|',MASS_PROPS(k+4,1),VOLUME_SUM(k+4),POWER_PROPS(k+4));
        fprintf('\n|                                 |                            |                            |
|');
        fprintf('\n|---------------------------------|----------------------------|----------------------------|----------
----------------|');
```

```
        k = k + 5;
    end
    if strcmpi(SENSOR_DATA{i,1},'ACCELERATION') == 1
        fprintf('\n|                                    |                          |                          |
|');
        fprintf('\n|     %17s          |        %10.4f          |        %10.4f          |        %10.4f
|',SENSOR_DATA{i,1},MASS_PROPS(k,1),VOLUME_SUM(k),POWER_PROPS(k));
        fprintf('\n|                                    |                          |                          |
|');
        fprintf('\n|------------------------------------|--------------------------|--------------------------|-----------
----------------|');
        k = k + 1;
    end
    if strcmpi(SENSOR_DATA{i,1},'ACOUSTICS') == 1
        fprintf('\n|                                    |                          |                          |
|');
        fprintf('\n|     %17s          |        %10.4f          |        %10.4f          |        %10.4f
|',SENSOR_DATA{i,1},MASS_PROPS(k,1),VOLUME_SUM(k),POWER_PROPS(k));
        fprintf('\n|                                    |                          |                          |
|');
        fprintf('\n|------------------------------------|--------------------------|--------------------------|-----------
----------------|');
        k = k + 1;
    end
    if strcmpi(SENSOR_DATA{i,1},'EM FIELD') == 1
        fprintf('\n|                                    |                          |                          |
|');
        fprintf('\n|     %17s          |        %10.4f          |        %10.4f          |        %10.4f
|',SENSOR_DATA{i,1},MASS_PROPS(k,1),VOLUME_SUM(k),POWER_PROPS(k));
        fprintf('\n|              Handheld Unit         |        %10.4f          |        %10.4f          |        %10.4f
|',MASS_PROPS3(1),VOLUME_SUM3,POWER_PROPS3);
        fprintf('\n|                                    |                          |                          |
|');
        fprintf('\n|------------------------------------|--------------------------|--------------------------|-----------
----------------|');
        k = k + 1;
    end
    if strcmpi(SENSOR_DATA{i,1},'GAS ANALYSIS') == 1
        fprintf('\n|                                    |                          |                          |
|');
        fprintf('\n|     %17s          |        %10.4f          |        %10.4f          |        %10.4f
|',SENSOR_DATA{i,1},MASS_PROPS(k,1),VOLUME_SUM(k),POWER_PROPS(k));
        fprintf('\n|                                    |                          |                          |
|');
        fprintf('\n|------------------------------------|--------------------------|--------------------------|-----------
----------------|');
        k = k + 1;
    end
    if strcmpi(SENSOR_DATA{i,1},'INCLINATION') == 1
        fprintf('\n|                                    |                          |                          |
|');
        fprintf('\n|     %17s          |        %10.4f          |        %10.4f          |        %10.4f
|',SENSOR_DATA{i,1},MASS_PROPS(k,1),VOLUME_SUM(k),POWER_PROPS(k));
```

```
        fprintf('\n|                                    |                          |                          |
|');
        fprintf('\n|------------------------------------|--------------------------|--------------------------|-----------
----------------|');
        k = k + 1;
    end
    if strcmpi(SENSOR_DATA{i,1},'NEPHELOMETRY') == 1
        fprintf('\n|                                    |                          |                          |
|');
        fprintf('\n|     %17s           |        %10.4f          |        %10.4f           |        %10.4f
|',SENSOR_DATA{i,1},MASS_PROPS(k,1),VOLUME_SUM(k),POWER_PROPS(k));
        fprintf('\n|                                    |                          |                          |
|');
        fprintf('\n|------------------------------------|--------------------------|--------------------------|-----------
----------------|');
        k = k + 1;
    end
    if strcmpi(SENSOR_DATA{i,1},'OPTICS') == 1
        fprintf('\n|                                    |                          |                          |
|');
        fprintf('\n|     %17s           |        %10.4f          |        %10.4f           |        %10.4f
|',SENSOR_DATA{i,1},MASS_PROPS(k,1),VOLUME_SUM(k),POWER_PROPS(k));
        fprintf('\n|                                    |                          |                          |
|');
        fprintf('\n|------------------------------------|--------------------------|--------------------------|-----------
----------------|');
        k = k + 1;
    end
    if strcmpi(SENSOR_DATA{i,1},'RADIATION') == 1
        fprintf('\n|                                    |                          |                          |
|');
        fprintf('\n|     %17s           |        %10.4f          |        %10.4f           |        %10.4f
|',SENSOR_DATA{i,1},MASS_PROPS(k,1),VOLUME_SUM(k),POWER_PROPS(k));
        fprintf('\n|                                    |                          |                          |
|');
        fprintf('\n|------------------------------------|--------------------------|--------------------------|-----------
----------------|');
        k = k + 1;
    end
    if strcmpi(SENSOR_DATA{i,1},'REFRACTION') == 1
        fprintf('\n|                                    |                          |                          |
|');
        fprintf('\n|     %17s           |        %10.4f          |        %10.4f           |        %10.4f
|',SENSOR_DATA{i,1},MASS_PROPS(k,1),VOLUME_SUM(k),POWER_PROPS(k));
        fprintf('\n|                                    |                          |                          |
|');
        fprintf('\n|------------------------------------|--------------------------|--------------------------|-----------
----------------|');
        k = k + 1;
    end
    if strcmpi(SENSOR_DATA{i,1},'DATA PROCESSING') == 1
        fprintf('\n|                                    |                          |                          |
|');
```

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

fprintf('\n|====================================|==========================|============================|==================
=========|');
fprintf('\n|      SENSOR PAYLOAD TOTAL          |        %10.4f          |        %10.4f             |        %10.4f
|',TOTAL_MASS,TOTAL_VOLUME,POWER_TOTAL);
fprintf('\n|====================================|==========================|============================|==================
=========|');

%  PRINT OUT CASE RESULTS FOR MASS VOLUME AND POWER LIMITS
fprintf('\n\n\n   ***   CASE LIMITS SUMMARY   ***');
fprintf('\n*****************************************************************');
fprintf('\n   CASE MASS LIMIT:     %s',MASS_FLAG);
for ln = 1:length(MASS_FLAG_STR)
    fprintf('\n\t\t\t\t\t   %s',MASS_FLAG_STR{ln});
end
fprintf('\n   CASE VOLUME LIMIT:   %s',VOLUME_FLAG);
for ln = 1:length(VOLUME_FLAG_STR)
    fprintf('\n\t\t\t\t\t   %s',VOLUME_FLAG_STR{ln});
end
fprintf('\n   CASE POWER LIMIT:    %s',POWER_FLAG);
for ln = 1:length(POWER_FLAG_STR)
    fprintf('\n\t\t\t\t\t   %s',POWER_FLAG_STR{ln});
end


cd ..;    %  Return to Case Working Directory

%  Create ISSPO SUMMARY RESULTS FILE
% ISSPO SUMMARY FILE ID
FISSPO = fopen('ISSPO_SUMMARY.txt','w+');



fprintf(FISSPO, '\n   ********************************************************************************************');
fprintf(FISSPO, '\n   ***   I N - S I T U   S E N S O R   P A Y L O A D   O P T I M I Z A T I O N   T O O L   ***');
fprintf(FISSPO, '\n   ********************************************************************************************');
fprintf(FISSPO, '\n\n***   P R O G R A M   I N P U T S   ***');
fprintf(FISSPO, '\n  User Input Filename:        %s', InputFile');
fprintf(FISSPO, '\n  Planetary Body Name:        %s', BODY);
fprintf(FISSPO, '\n  Unit System:                %s', UNITS);
fprintf(FISSPO, '\n  Print Summary Option:       %s', PRINTFLG);

fprintf(FISSPO, '\n\n  Payload Mass Limit:      %10.3f   %s', MASS_LIMIT, MassUnit);
fprintf(FISSPO, '\n  Payload Volume Limit:    %10.4f   %s', VOLUME_LIMIT, VolUnit);
fprintf(FISSPO, '\n  Payload Power Limit:     %10.3f   %s', POWER_LIMIT, PowerUnit);
```

```
fprintf(FISSPO, '\n\n*********************************************************************************\n\n');


fprintf(FISSPO, '\n   SENSOR PACKAGES DESIGNED:');
fprintf(FISSPO, '\n-------------------------------------------------------------------------------');

for i = 1:size(SENSOR_DATA,1)
    fprintf(FISSPO, '\n     %s', SENSOR_DATA{i,1});
    if strcmpi(SENSOR_DATA{i},'ATMOSPHERE') == 1
        fprintf(FISSPO, '\n        - Temperature');
        fprintf(FISSPO, '\n        - Pressure');
        fprintf(FISSPO, '\n        - Wind Velocity');
        fprintf(FISSPO, '\n        - Humidity');
        fprintf(FISSPO, '\n        - Density');
    end
    if strcmpi(SENSOR_DATA{i},'OPTICS') == 1
        fprintf(FISSPO, '\n        - %s', SENSOR_DATA{i,2});
        for r = 3:size(SENSOR_DATA,2)
            fprintf(FISSPO, '\n        - %s', SENSOR_DATA{i,r});
        end
    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(FISSPO, '\n        - %s', SENSOR_DATA{i,r});
            else
                break;
            end
        end
    end
end


%   Print  Mass Volume Breakdown Summary
k = 1;
fprintf(FISSPO, '\n\n\n                          ***   P A Y L O A D   S E N S O R   B R E A K D O W N   ***');
fprintf(FISSPO,
'\n\n|==================================|============================|==============================|============================
===|');
fprintf(FISSPO, '\n|             SENSOR               |            MASS            |            VOLUME            |
POWER            |');
fprintf(FISSPO,
'\n|==================================|============================|==============================|============================
=|');
fprintf(FISSPO, '\n|----------------------------------|           %4s             |           %6s             |           %6s
|', MassUnit,VolUnit,PowerUnit);
fprintf(FISSPO,
'\n|==================================|============================|==============================|============================
=|');
for i = 1:size(SENSOR_DATA,1)
    if strcmpi(SENSOR_DATA{i,1},'ATMOSPHERE') == 1
        fprintf(FISSPO, '\n|                                  |                            |                              |
|');
```

```
        fprintf(FISSPO, '\n|     %17s            |                            |                              |
|',SENSOR_DATA{k,1} );
        fprintf(FISSPO, '\n|         - Temperature             |        %10.4f          |        %10.4f            |
%10.4f          |',MASS_PROPS(k,1),VOLUME_SUM(k),POWER_PROPS(k));
        fprintf(FISSPO, '\n|         - Pressure                |        %10.4f          |        %10.4f            |
%10.4f          |',MASS_PROPS(k+1,1),VOLUME_SUM(k+1),POWER_PROPS(k+1));
        if strcmpi(WV_SENS_TYPE,'ANEMOMETER') == 1
            fprintf(FISSPO, '\n|         - Wind Velocity Probe     |        %10.4f          |        %10.4f            |
%10.4f          |',MASS_PROPS(k+2,1),VOLUME_SUM(k+2),POWER_PROPS(k+2));
            fprintf(FISSPO, '\n|                 Handheld Unit     |        %10.4f          |        %10.4f            |
%10.4f          |',MASS_PROPS2(1),VOLUME_SUM2,POWER_PROPS2);
        end
        if strcmpi(WV_SENS_TYPE,'DOPPLER') == 1
            fprintf(FISSPO, '\n|         - Wind Velocity Probe     |        %10.4f          |        %10.4f            |
%10.4f          |',2*MASS_PROPS(k+2,1),VOLUME_SUM(k+2),POWER_PROPS(k+2));
        end
        fprintf(FISSPO, '\n|         - Humidity                |        %10.4f          |        %10.4f            |
%10.4f          |',MASS_PROPS(k+3,1),VOLUME_SUM(k+3),POWER_PROPS(k+3));
        fprintf(FISSPO, '\n|         - Density                 |        %10.4f          |        %10.4f            |
%10.4f          |',MASS_PROPS(k+4,1),VOLUME_SUM(k+4),POWER_PROPS(k+4));
        fprintf(FISSPO, '\n|                                   |                            |                              |
|');
        fprintf(FISSPO, '\n|-----------------------------------|----------------------------|------------------------------|----
------------------------|');
        k = k + 5;
    end
    if strcmpi(SENSOR_DATA{i,1},'ACCELERATION') == 1
        fprintf(FISSPO, '\n|                                   |                            |                              |
|');
        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|                                   |                            |                              |
|');
        fprintf(FISSPO, '\n|-----------------------------------|----------------------------|------------------------------|----
------------------------|');
        k = k + 1;
    end
    if strcmpi(SENSOR_DATA{i,1},'ACOUSTICS') == 1
        fprintf(FISSPO, '\n|                                   |                            |                              |
|');
        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|                                   |                            |                              |
|');
        fprintf(FISSPO, '\n|-----------------------------------|----------------------------|------------------------------|----
------------------------|');
        k = k + 1;
    end
    if strcmpi(SENSOR_DATA{i,1},'EM FIELD') == 1
        fprintf(FISSPO, '\n|                                   |                            |                              |
|');
        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|              Handheld Unit       |        %10.4f          |        %10.4f            |
%10.4f           |',MASS_PROPS3(1),VOLUME_SUM3,POWER_PROPS3);
        fprintf(FISSPO, '\n|                                  |                        |                          |
|');
        fprintf(FISSPO, '\n|----------------------------------|------------------------|--------------------------|----
------------------------|');
        k = k + 1;
    end
    if strcmpi(SENSOR_DATA{i,1},'GAS ANALYSIS') == 1
        fprintf(FISSPO, '\n|                                  |                        |                          |
|');
        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|                                  |                        |                          |
|');
        fprintf(FISSPO, '\n|----------------------------------|------------------------|--------------------------|----
------------------------|');
        k = k + 1;
    end
    if strcmpi(SENSOR_DATA{i,1},'INCLINATION') == 1
        fprintf(FISSPO, '\n|                                  |                        |                          |
|');
        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|                                  |                        |                          |
|');
        fprintf(FISSPO, '\n|----------------------------------|------------------------|--------------------------|----
------------------------|');
        k = k + 1;
    end
    if strcmpi(SENSOR_DATA{i,1},'NEPHELOMETRY') == 1
        fprintf(FISSPO, '\n|                                  |                        |                          |
|');
        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|                                  |                        |                          |
|');
        fprintf(FISSPO, '\n|----------------------------------|------------------------|--------------------------|----
------------------------|');
        k = k + 1;
    end
    if strcmpi(SENSOR_DATA{i,1},'OPTICS') == 1
        fprintf(FISSPO, '\n|                                  |                        |                          |
|');
        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|                                  |                        |                          |
|');
        fprintf(FISSPO, '\n|----------------------------------|------------------------|--------------------------|----
------------------------|');
        k = k + 1;
    end
    if strcmpi(SENSOR_DATA{i,1},'RADIATION') == 1
```

```
        fprintf(FISSPO, '\n|                                    |                            |                            |
|');
        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|                                    |                            |                            |
|');
        fprintf(FISSPO, '\n|------------------------------------|----------------------------|----------------------------|----
------------------------|');
        k = k + 1;
    end
    if strcmpi(SENSOR_DATA{i,1},'REFRACTION') == 1
        fprintf(FISSPO, '\n|                                    |                            |                            |
|');
        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|                                    |                            |                            |
|');
        fprintf(FISSPO, '\n|------------------------------------|----------------------------|----------------------------|----
------------------------|');
        k = k + 1;
    end
    if strcmpi(SENSOR_DATA{i,1},'DATA PROCESSING') == 1
        fprintf(FISSPO, '\n|                                    |                            |                            |
|');
        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|                                    |                            |                            |
|');
        fprintf(FISSPO, '\n|------------------------------------|----------------------------|----------------------------|----
------------------------|');
        k = k + 1;
    end
end

fprintf(FISSPO,
'\n|====================================|============================|============================|============================
=|');
fprintf(FISSPO, '\n|      SENSOR PAYLOAD TOTAL          |        %10.4f          |        %10.4f           |        %10.4f
|',TOTAL_MASS,TOTAL_VOLUME,POWER_TOTAL);
fprintf(FISSPO,
'\n|====================================|============================|============================|============================
=|');

%  PRINT OUT CASE RESULTS FOR MASS VOLUME AND POWER LIMITS
fprintf(FISSPO, '\n\n\n   ***   CASE LIMITS SUMMARY   ***');
fprintf(FISSPO, '\n*********************************************************************');
fprintf(FISSPO, '\n   CASE MASS LIMIT:     %s',MASS_FLAG);
for ln = 1:length(MASS_FLAG_STR)
    fprintf(FISSPO, '\n\t\t\t\t\t   %s',MASS_FLAG_STR{ln});
end
fprintf(FISSPO, '\n   CASE VOLUME LIMIT:   %s',VOLUME_FLAG);
for ln = 1:length(VOLUME_FLAG_STR)
    fprintf(FISSPO, '\n\t\t\t\t\t   %s',VOLUME_FLAG_STR{ln});
end
```

```
fprintf(FISSPO, '\n   CASE POWER LIMIT:    %s',POWER_FLAG);
for ln = 1:length(POWER_FLAG_STR)
    fprintf(FISSPO, '\n\t\t\t\t\t   %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,'\n\n\n');

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,'\n\n\n');
            fclose(FIDT);

            %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,'\n\n\n');
            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,'\n\n\n');
            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,'\n\n\n');
            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,'\n\n\n');
            fclose(FIDD);

        case 'ACCELERATION'
```

```
    %Reads and copies ACCELERATION SUMMARY to ISSPO Summary File
    FIDACC = fopen('ACCELEROMETER_SENSOR_Summary.txt','r');
    DataACC = fread(FIDACC);
    fwrite(FISSPO,DataACC,'char');
    fprintf(FISSPO,'\n\n\n');
    fclose(FIDACC);
case 'ACOUSTICS'
    %Reads and copies ACOUCTICS SUMMARY to ISSPO Summary File
    FIDACS = fopen('ACOUSTIC_SENSOR_Summary.txt','r');
    DataACS = fread(FIDACS);
    fwrite(FISSPO,DataACS,'char');
    fprintf(FISSPO,'\n\n\n');
    fclose(FIDACS);
case 'EM FIELD'
    %Reads and copies AC FIELD SUMMARY to ISSPO Summary File
    FIDACF = fopen('AC_FIELD_SENSOR_Summary.txt','r');
    DataACF = fread(FIDACF);
    fwrite(FISSPO,DataACF,'char');
    fprintf(FISSPO,'\n\n\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\n\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\n\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\n\n');
    fclose(FIDNPH);
case 'OPTICS'
    %Reads and copies IMAGING SUMMARY to ISSPO Summary File
    FIDIMG = fopen('IMAGING_SENSOR_Summary.txt','r');
    DataIMG = fread(FIDIMG);
    fwrite(FISSPO,DataIMG,'char');
    fprintf(FISSPO,'\n\n\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\n\n');
```

```
            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\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');
            fprintf(FISSPO,'\n\n\n');
            fclose(FIDDSP);
    otherwise
        fprintf('\n\nSENSOR_DATA(%d):  %s\n', i, SENSOR_DATA{i,1} );
        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\n*****************************************');
fprintf('\n   ISSPO DESIGN TOOL FINISHED!!');
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.
```

```
%  Some of the values are based on the Unit system chosen and are loaded
%  in separate sections.

function Constants(UNITS)

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

    Guniv = 6.67E-11;
    Guniv_lbl = 'Nm^2/kg^2';
    m2m = 1;
    m2m_lbl = 'm/m';
    cm2m = 100;
    cm2m_lbl = 'cm/m';
    mm2m = 1000;
    mm2m_lbl = 'mm/m';
    gr2kg = 1000;
    gr2kg_lbl = 'gram/kg';
    kg2kg = 1;
    kg2kg_lbl = 'kg/kg';
    cm3_to_m3 = 1000000;
    cm3_to_m3_lbl = 'cm^3/m^3';
    kgm3_to_gcm3 = 0.001;
    kgm3_to_gcm3_lbl = 'kg/m^3 / g/cm^3';
    mA2A = 1000;
    mA2A_lbl = 'mA/A';
    mW2W = 1000;
    mW2W_lbl = 'mW/W';
    Press_Cnvt = 1000;
    Press_Cnvt_lbl = 'Pa to kPa';
    Temp_C2K = 273.15;
    Temp_C2K_lbl = '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;
    Guniv_lbl = 'lbf*ft^2/lb^2';
    ft2mi = 5280;
    ft2mi_lbl = 'ft/mi';
    in2ft = 12;
    in2ft_lbl = 'in/ft';
    oz2lbm = 16;
    oz2lbm_lbl = 'oz/lbm';
```

```
    slug2lbm = 32.2;
    slug2lbm_lbl = 'slug/lbm';
    in3_to_ft3 = 1728;
    in3_to_ft3_lbl = 'in^3/ft^3';
    lbft3_to_lbft3 = 1;
    lbft3_to_lbft3_lbl = 'lb/ft^3 / lb/ft^3'
    mA2A = 1000;
    mA2A_lbl = 'mA/A';
    BTU_hr2W = 3.412141;
    BTU_hr2W_lbl = 'BTU/hr/W';
    Press_Cnvt = 1;
    Press_Cnvt_lbl = 'psi to psi';
    Temp_F2R = 459.67;
    Temp_F2R_lbl = 'deg F to deg R';
    MassUnit = 'lbm';
    VolUnit = 'ft^3';
    PowerUnit = 'BTU''s/hr';
    DistUnit = 'mi';
    GravUnit = 'ft/s^2';
    TempUnit = 'deg F';
    PressureUnit = 'psi';
    DensityUnit = 'lbm/ft^3';
    VelUnit = 'ft/s';
    GMUnit = 'ft^3/s^2';
    SolIradUnit = 'BTUs/h*ft^2';
    MagFieldUnit = 'mGauss';
else
    fprintf('\n\nERROR: Unknown Unit Type!');
    fprintf('\nTerminating Program. . .');
end


save('Constants_DB.mat');
```

D3. Data Type Verifier

```
%   function DataTypeVerifier(DataArray)
%   ISSPO 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('\n\nSENSOR_DATA{%d,1}:  %s\n', 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('\n\nSENSOR_DATA{%d,%d}:  %s\n', i, pos+1, DataArray{i,pos+1} );
                            DataTypeERR_FLG = 15;
                            return;
                        end
                    else
                        DataTypeERR_FLG = 14;
                        return;
                    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('\n\nSENSOR_DATA{%d,%d}:  %s\n', i, pos+1, DataArray{i,pos+1} );
                            DataTypeERR_FLG = 22;
                            return;
                        end
                    else
                        DataTypeERR_FLG = 21;
                        return;
                    end
                end
```

```
                if strcmpi(DataArray{i,pos},'DENSITY') == 1
                    if size(DataArray,2) >= pos+1
                        if (strcmpi(DataArray{i,pos+1},'GAS') ~= 1) && (strcmpi(DataArray{i,pos+1},'LIQUID') ~= 1)
                            fprintf('\n\nSENSOR_DATA{%d,%d}:  %s\n', i, pos+1, DataArray{i,pos+1} );
                            DataTypeERR_FLG = 26;
                            return;
                        end
                    else
                        DataTypeERR_FLG = 25;
                        return;
                    end
                end
            end
        end
    end


    if strcmpi(DataArray{i,1},'GAS ANALYSIS') == 1
        if size(DataArray,2) == 1
            fprintf('\n\nSENSOR_DATA{%d,1}:  %s\n', 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('\n\nSENSOR_DATA{%d,3}:  %s\n\n', i, DataArray{i,3} );
                    DataTypeERR_FLG = 18;
                    return;
                end
            end
        else
            fprintf('\n\nSENSOR_DATA{%d,2}:  %s\n\n', i, DataArray{i,2} );
            DataTypeERR_FLG = 16;
            return;
        end
    end

    if strcmpi(DataArray{i,1},'OPTICS') == 1
        if size(DataArray,2) == 1
            fprintf('\n\nSENSOR_DATA{%d,1}:  %s\n', i, DataArray{i,1} );
            DataTypeERR_FLG = 2;
            return;
        end

        if size(DataArray,2) == 2
            fprintf('\n\nSENSOR_DATA{%d,1}:  %s\n', i, DataArray{i,2} );
```

```
            DataTypeERR_FLG = 3;
            return;
        end

        if size(DataArray,2) == 3
            fprintf('\n\nSENSOR_DATA{%d,1}:  %s\n', i, DataArray{i,3} );
            DataTypeERR_FLG = 4;
            return;
        end

        if (strcmpi(DataArray{i,2},'CAMERA')==1) || (strcmpi(DataArray{i,2},'ARRAY')==1) ||
(strcmpi(DataArray{i,2},'LINEAR')==1)
        else
            fprintf('\n\nSENSOR_DATA{%d,2}:  %s\n\n', 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\nSENSOR_DATA{%d,3}:  %s\n\n', 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) ||
(strcmpi(DataArray{i,r},'UV')==1)...
                        || (strcmpi(DataArray{i,r},'NIR')==1) ||  (strcmpi(DataArray{i,r},'MICRO')==1)
                else
                    fprintf('\n\nSENSOR_DATA{%d,%d}:  %s\n\n', i, r, DataArray{i,r} );
                    DataTypeERR_FLG = 7;
                    return;
                end
            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\nSENSOR_DATA{%d,1}:  %s\n', 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('\n\nSENSOR_DATA{%d,%d}:  %s\n\n', i, r, DataArray{i,r} );
                    DataTypeERR_FLG = 9;
                    return;
                end
            else
                %  if N/A is Encountered in DataArray, Stops Reading in Values
                break;
            end
        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('\n\nSENSOR_DATA{%d,1}:  %s\n', 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('\n\nSENSOR_DATA{%d,2}:  %s\n\n', 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('\n\nSENSOR_DATA{%d,1}:  %s\n', i, DataArray{i,1} );
            DataTypeERR_FLG = 12;
            return;
        end

        if size(DataArray,2) < 3
            fprintf('\n\nSENSOR_DATA{%d,1}:  %s\n', i, DataArray{i,1} );
            DataTypeERR_FLG = 19;
            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('\n\nSENSOR_DATA{%d,2}:  %s\n\n', i, DataArray{i,2} );
```

```
            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('\n\nSENSOR_DATA{%d,3}:  %s\n\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('\n\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('\n\nSENSOR_DATA{%d,2}:  %s\n\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('\n\nSENSOR_DATA{%d,1}:  %s\n', i, DataArray{i,1} );
            DataTypeERR_FLG = 27;
            return;
        end

        if size(DataArray,2) < 3
            fprintf('\n\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('\n\nSENSOR_DATA{%d,2}:  %s\n\n', i, DataArray{i,2} );
            DataTypeERR_FLG = 29;
            return;
        end
```

```
        if isnumeric(DataArray{i,3}) == 0
            fprintf('\n\nSENSOR_DATA{%d,3}:  %s\n\n', i, DataArray{i,3} );
            DataTypeERR_FLG = 30;
            return;
        end

    end   %  Closes DIGITAL SIGNAL PROCESSING Sensor Definition Area


end


%  return;
```

D4. Planetary Database

```
% function PlanetaryDatabase(Body, Units, Print)
%
%  Developed by:  Keith Schreck
%                 Mechanical and Aerospace Engineering
%                 San Jose State University
%  Date:          Fall 2007
%
%  This Program is a Planetary Database Model for the ISSPO Tool.  Given a
%  Planetary Name and a Unit System (S.I. or British) it will return the
%  associated planetary properties.

%  Program can be evaluated in two modes, either part of the ISSPO Tool or
%  as a standalone tool to return Planetary Body data.  The Program is
%  either called from the MatLAB command line via:
%      PlanetaryDataBase('Planet Name', 'Unit System', 'Print')
%    where, Planet Name is it name of the object under scrutiny, and
%      Unit System, is the desired Unit System either 'SI', or 'British'.
%      Print is 'Y', or 'N' whether or not to print results to the screen.
%      when running ISSPO, Print is 'N' to suppress intermediate printouts,
%      but when running PlanetaryDatabase as a standalone program, the
%      user will probably want the results printed to the screen.
%
%  If these values are not input at the command line the program will then
%  prompt the user for the proper inputs before continuing.

%   References
%     http://nssdc.gsfc.nasa.gov/planetary/planetfact.html

function PlanetaryDatabase(varargin)

%  Setup for Program
%clc;
%clear all;
close all;
format compact;

%  Define Program Input variables
```

```
Print = 'N';
BODY = 'Default';
UNITS = 'Default';
Temp_C2K = 273.15;
Temp_F2R = 459.67;

%  Checks Input number of variable to see if Properly Formatted
if (nargin == 3)
    %fprintf('\n\n  Success!!\n');
    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 {'Metis', 'metis'};        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('\nERROR:  Unknown Planetary Object in Planetary Database Program from Preloaded Inputs!!\n');
    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('\n*                                                                               *');
    fprintf('\n*          P L A N E T A R Y    D A T A B A S E    P R O G R A M               *');
    fprintf('\n*                                                                               *');
    fprintf('\n*********************************************************************************\n\n');
elseif( (nargin == 3) && strcmpi(Print,'N') == 1)
```

```
    %  Does not Print Anything does Not exit
else
    fprintf('\n\n   ***ERROR:  Invalid Number of Input Parameters');
    fprintf('\n    Terminating Program Operation . . .');
    return;
end


if (nargin == 3) && (strcmpi(Print,'Y') == 1)
    fprintf('\n   *** Loading Input Program Values ***\n\n');
end

if nargin == 0
    fprintf('\n*** Enter Program Values ***');
    fprintf('\nPlanetary Bodies Available to Analyze:');
    fprintf('\n 0 - Sun   \n 1 - Mercury   \n 2 - Venus   \n 3 - Earth   \n   3.1 - Moon   \n 4 - Mars');
    fprintf('\n 5 - Jupiter   \n   Jovian Satellites   \n     5.1 - Io   \n     5.2 - Europa   ');
    fprintf('\n     5.3 - Ganymede   \n     5.4 - Callisto   \n     5.5 - Metis   \n     5.6 - Adrastea');
    fprintf('\n     5.7 - Amalthea   \n     5.8 - Thebe');
    fprintf('\n 6 - Saturn    \n   Saturnian Satellites   \n     6.1 - Mimas   \n     6.2 - Enceladus');
    fprintf('\n     6.3 - Tethys   \n     6.4 - Dione   \n     6.5 - Rhea   \n     6.6 - Titan');
    fprintf('\n     6.7 - Hyperion   \n     6.8 - Iapetus');
    fprintf('\n 7 - Uranus   \n 8 - Neptune   \n 9 - Pluto');

    while strcmpi(BODY,'Default') == 1
        SwitchCase = input('\n\nEnter 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';
    SolIradUnit = 'W/m^2';
elseif strcmpi(UNITS,'British') == 1
    MassUnit = 'lbm';
    VolUnit = 'ft^3';
    DistUnit = 'mi';
    GravUnit = 'ft/s^2';
```

```
    TempUnit = 'deg F';
    PressureUnit = 'psi';
    DensityUnit = 'lbm/ft^3';
    VelUnit = 'ft/s';
    GMUnit = 'ft^3/s^2';
    SolIradUnit = 'BTUs/h*ft^2';
else
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%   Planetary Data Base Created Based on un Unit System
%%  2 - D Matrix  Planet_DB(BODY_LOC, Property)

if strcmpi(UNITS,'SI') == 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            kg        10^10 m^3        m                 m              m/s^2        deg C        Pascals
kg/m^3         N/A            m/s          km^3/s^2         N/A          N/A       V(1,0)       W/m^2          deg K        kg/m^3
mGauss        I/MR^2     x10^-6      SATELLITES    ]
    Planet_DB = {   'SUN'         1.9891E+30  1.41200E+17   6.9600E+08     6.9600E+08   2.740E+02    5778.00      8.680E+01
1.4080E+03   5.0000E-05    6.1760E+05     1.3271E+11   0.000E+00     0.000E+00  -26.74    0.0000E+00     5.778E+03   0.000E+00
0.00       5.900E-02  0.00000E+00       0          ;% Sun
                    'MERCURY'     3.3022E+23  6.08300E+09   2.4397E+06     2.4397E+06   3.700E+00     167.00      1.000E-10
5.4270E+03   0.0000E+00    4.3000E+03     2.2030E+04   1.190E-01     1.060E-01   -0.42    9.1266E+03     4.425E+02   0.000E+00
0.00       3.300E-01  6.00000E+01       0          ;% Mercury
                    'VENUS'       4.8690E+24  9.28430E+10   6.0518E+06     6.0518E+06   8.870E+00     464.00      9.200E+06
5.2430E+03   0.0000E+00    1.0360E+04     3.2490E+05   7.500E-01     6.500E-01   -4.40    2.6139E+03     2.317E+02   6.440E+01
0.00       3.300E-01  4.45800E+00       0          ;% Venus
                                   'EARTH'       5.9742E+24  1.08321E+11   6.3781E+06     6.3781E+06   9.800E+00       15.00
1.013E+05    5.5150E+03    3.3500E-03    1.1200E+04     3.9860E+05   3.060E-01    3.670E-01   -3.86    1.3676E+03     2.543E+02
1.225E+00      310.00        3.308E-01  1.08263E+03       1          ;% Earth
                                   'MOON'        7.3490E+22  2.19580E+09   1.7381E+06     1.7360E+06   1.620E+00      -23.15
3.000E-10    3.3500E+03    1.2000E-03    2.3800E+03     4.9000E+03   1.100E-01    1.200E-01   +0.21    1.3676E+03     2.745E+02
0.000E+00       0.00         3.940E-01  2.02700E+02       0          ;% Moon
                                   'MARS'        6.4185E+23  1.63180E+10   3.3970E+06     3.3750E+06   3.710E+00      -63.00
6.360E+02    3.9330E+03    6.4800E-03    5.0300E+03     4.2830E+04   2.500E-01    1.500E-01   -1.52    5.8920E+02     2.101E+02
0.000E+00       0.00         3.660E-01  1.96045E+03       2          ;% Mars
                                   'JUPITER'     1.8986E+27  1.43128E+14   7.1492E+07     6.6854E+07   2.479E+01     -108.00
1.000E+08    1.3260E+03    6.4870E-02    5.9500E+04     1.2669E+08   3.430E-01    5.200E-01   -9.40    5.0500E+01     1.100E+02
0.000E+00     4280.00        2.540E-01  1.47360E+04      63          ;% Jupiter
                                   'IO'          8.9320E+22  2.53191E+09   1.8216E+06     1.8216E+06   0.000E+00        0.00
0.000E+00    3.5300E+03    0.0000E+00    0.0000E+00     0.0000E+00   0.000E+00    6.200E-01    0.00    0.0000E+00     0.000E+00
0.000E+00       0.00         0.000E+00  0.00000E+00       0          ;%   Io
                    'EUROPA'      4.8000E+22  1.59269E+09   1.5608E+06     1.5608E+06   0.000E+00       0.00      0.000E+00
3.0100E+03   0.0000E+00    0.0000E+00     0.0000E+00   0.000E+00     6.800E-01    0.00    0.0000E+00     0.000E+00   0.000E+00
0.00       0.000E+00  0.00000E+00       0          ;%   Europa
                    'GANYMEDE'    1.4819E+23  7.63045E+09   2.6312E+06     2.6312E+06   0.000E+00       0.00      0.000E+00
1.9400E+03   0.0000E+00    0.0000E+00     0.0000E+00   0.000E+00     4.400E-01    0.00    0.0000E+00     0.000E+00   0.000E+00
0.00       0.000E+00  0.00000E+00       0          ;%   Ganymede
                    'CALLISTO'    1.0759E+23  5.86546E+09   2.4103E+06     2.4103E+06   0.000E+00       0.00      0.000E+00
1.8300E+03   0.0000E+00    0.0000E+00     0.0000E+00   0.000E+00     1.900E-01    0.00    0.0000E+00     0.000E+00   0.000E+00
0.00       0.000E+00  0.00000E+00       0          ;%   Callisto
```

```
                    'METIS'      1.0000E+17  3.35103E+04   2.0000E+04     2.0000E+04   0.000E+00       0.00      0.000E+00
0.0000E+00   0.0000E+00    0.0000E+00    0.0000E+00   0.000E+00    6.000E-02    0.00    0.0000E+00     0.000E+00   0.000E+00
0.00      0.000E+00  0.00000E+00      0        ;%   Metis
                    'ADRASTEA'   2.0000E+16  4.35634E+02   1.3000E+04     8.0000E+03   0.000E+00       0.00      0.000E+00
0.0000E+00   0.0000E+00    0.0000E+00    0.0000E+00   0.000E+00    1.000E-01    0.00    0.0000E+00     0.000E+00   0.000E+00
0.00      0.000E+00  0.00000E+00      0        ;%   Adrastea
                    'AMALTHEA'   7.5000E+18  2.68385E+05   1.3100E+05     6.7000E+04   0.000E+00       0.00      0.000E+00
3.1000E+03   0.0000E+00    0.0000E+00    0.0000E+00   0.000E+00    9.000E-02    0.00    0.0000E+00     0.000E+00   0.000E+00
0.00      0.000E+00  0.00000E+00      0        ;%   Amalthea
                    'THEBE'      8.0000E+17  5.70199E+04   5.5000E+04     4.5000E+04   0.000E+00       0.00      0.000E+00
0.0000E+00   0.0000E+00    0.0000E+00    0.0000E+00   0.000E+00    5.000E-02    0.00    0.0000E+00     0.000E+00   0.000E+00
0.00      0.000E+00  0.00000E+00      0        ;%   Thebe
                  'SATURN'      5.6846E+26  8.27130E+13   6.0268E+07     5.4364E+07   1.044E+01    -139.15      1.000E+08
6.8700E+02   9.7960E-02    3.5500E+04    3.7931E+07   3.420E-01    4.700E-01   -8.88    1.4900E+01     8.110E+01   0.000E+00
220.00      2.100E-01  1.62980E+04     56        ;% Saturn
                    'MIMAS'      3.7900E+19  3.27736E+06   2.0900E+05     1.9100E+05   0.000E+00       0.00      0.000E+00
1.1500E+03   0.0000E+00    0.0000E+00    0.0000E+00   0.000E+00    6.000E-01    0.00    0.0000E+00     0.000E+00   0.000E+00
0.00      0.000E+00  0.00000E+00      0        ;%   Mimas
                    'ENCELADUS'  1.0800E+20  6.48921E+06   2.5600E+05     2.4500E+05   0.000E+00       0.00      0.000E+00
1.6100E+03   0.0000E+00    0.0000E+00    0.0000E+00   0.000E+00    1.000E+00    0.00    0.0000E+00     0.000E+00   0.000E+00
0.00      0.000E+00  0.00000E+00      0        ;%   Enceladus
                    'TETHYS'     6.1800E+20  6.23553E+07   5.3600E+05     5.2600E+05   0.000E+00       0.00      0.000E+00
9.6000E+02   0.0000E+00    0.0000E+00    0.0000E+00   0.000E+00    8.000E-01    0.00    0.0000E+00     0.000E+00   0.000E+00
0.00      0.000E+00  0.00000E+00      0        ;%   Tethys
                    'DIONE'      1.1000E+21  7.35619E+07   5.6000E+05     5.6000E+05   0.000E+00       0.00      0.000E+00
1.4700E+03   0.0000E+00    0.0000E+00    0.0000E+00   0.000E+00    7.000E-01    0.00    0.0000E+00     0.000E+00   0.000E+00
0.00      0.000E+00  0.00000E+00      0        ;%   Dione
                    'RHEA'       2.3100E+21  1.86796E+08   7.6400E+05     7.6400E+05   0.000E+00       0.00      0.000E+00
1.2300E+03   0.0000E+00    0.0000E+00    0.0000E+00   0.000E+00    7.000E-01    0.00    0.0000E+00     0.000E+00   0.000E+00
0.00      0.000E+00  0.00000E+00      0        ;%   Rhea
                    'TITAN'      1.3455E+23  7.15188E+09   2.5750E+06     2.5750E+06   1.352E+00    -179.45      1.467E+05
1.8800E+03   0.0000E+00    2.6390E+03    0.0000E+00   2.200E-01    2.200E-01    7.90    1.5044E+01     0.000E+00   5.500E+00
0.00      0.000E+00  0.00000E+00      0        ;%   Titan
                    'HYPERION'   5.5000E+18  1.22593E+06   1.8500E+05     1.1300E+05   0.000E+00       0.00      0.000E+00
5.7000E+02   0.0000E+00    0.0000E+00    0.0000E+00   0.000E+00    3.000E-01    0.00    0.0000E+00     0.000E+00   0.000E+00
0.00      0.000E+00  0.00000E+00      0        ;%   Hyperion
                    'IAPETUS'    1.8100E+21  1.55046E+08   7.1800E+05     7.1800E+05   0.000E+00       0.00      0.000E+00
1.0900E+03   0.0000E+00    0.0000E+00    0.0000E+00   0.000E+00    5.000E-01    0.00    0.0000E+00     0.000E+00   0.000E+00
0.00      0.000E+00  0.00000E+00      0        ;%   Iapetus
                                      'URANUS'     8.6832E+25  6.83300E+12   2.5559E+07     2.4973E+07   8.870E+00    -197.00
1.000E+08   1.2700E+03   2.2930E-02    2.1300E+04    5.7940E+06   3.000E-01    5.100E-01   -7.19    3.7100E+00     5.820E+01
0.000E+00      230.00      2.250E-01  3.34343E+03      27        ;% Uranus
                                      'NEPTUNE'    1.0243E+26  6.25400E+12   2.4764E+07     2.4341E+07   1.115E+01    -201.00
1.000E+08   1.6380E+03   1.7080E-02    2.3500E+04    6.8351E+06   2.900E-01    4.100E-01   -6.87    1.5100E+00     4.660E+01
0.000E+00      140.00      0.000E+00  3.41100E+03      13        ;% Neptune
                                      'PLUTO'      1.2500E+22  7.15000E+09   1.1950E+06     1.1950E+06   5.800E-01    -223.00
3.000E-01   1.7500E+03   0.0000E+00    1.2000E+03    8.3000E+02   5.000E-01    6.000E-01   -1.00    8.9000E-01     3.750E+01
0.000E+00        0.00      0.000E+00  0.00000E+00      3       };% Pluto

 % Properties  [ SEMIMAJOR-Axis  SIDEREAL-Period  TROPICAL-Period   PERIHELION      APHELION     SYNODIC-Per   MEAN-VEL
MAX-VEL     MIN-VEL     ORB-INCLIN    ECCENTRICITY    ROTATION-Per    LENGTH-DAY    OBLIQUITY to Orbit     ]
 %    Units                m             E. Days          E. Days          m              m          E. Days        m/s
m/s          m/s            deg             N/A          E. Hours       E. Hours          deg             ]
```

```
    Planet_DB2 = {  0.00000E+00     0.000000E+00     0.000000E+00    0.00000E+00    0.00000E+00   2.5449e+01   1.940E+04
0.000E+00   0.000E+00    0.0000E+00     0.000E+00     0.000000E+00    0.00000E+00       7.2500E+00      ;% Sun
                    5.79100E+10     8.796900E+01     8.796800E+01    4.60000E+10    6.98200E+10   1.1588E+02   4.787E+04
5.898E+04   3.886E+04    7.0000E+00     2.056E-01     1.407600E+03    4.22260E+03       1.0000E-02      ;% Mercury
                    1.08210E+11     2.247010E+02     2.246950E+02    1.07480E+11    1.08940E+11   5.8392E+02   3.502E+04
3.526E+04   3.479E+04    3.3900E+00     6.700E-02     -5.832500E+03    2.80200E+03       1.7736E+02      ;% Venus
                                    1.49600E+11     3.652560E+02     3.652420E+02    1.47090E+11    1.52100E+11   0.0000E+00
2.978E+04    3.029E+04   2.929E+04     0.0000E+00    1.670E-02       2.393450E+01    2.40000E+01       2.3450E+01      ;% Earth
                                3.84400E+08     0.000000E+00     0.000000E+00    3.63300E+08    4.05500E+08   2.9530E+01
1.023E+03    1.076E+03   9.640E+02     5.1450E+00    5.490E-02       6.557280E+02    0.00000E+00       6.6800E+00      ;% Moon
                                    2.27920E+11     6.869800E+02     6.869730E+02    2.06620E+11    2.49230E+11   7.7994E+02
2.413E+04    2.650E+04   2.197E+04     1.8500E+00    9.350E-02       2.462290E+01    2.46597E+01       2.5190E+01      ;% Mars
                                    7.78570E+11     4.332589E+03     4.330595E+03    7.40520E+11    8.16620E+11   3.9888E+02
1.307E+04    1.372E+04   1.244E+04     1.3040E+00    4.890E-02       9.925000E+00    9.92590E+00       3.1300E+00      ;% Jupiter
                    4.21600E+05     0.000000E+00     0.000000E+00    0.00000E+00    0.00000E+00   1.7691E+00   0.000E+00
0.000E+00   0.000E+00    4.0000E-02     4.000E-03     4.245931E+01    4.24593E+01       0.0000E+00      ;%   Io
                    6.70900E+05     0.000000E+00     0.000000E+00    0.00000E+00    0.00000E+00   3.5512E+00   0.000E+00
0.000E+00   0.000E+00    4.7000E-01     1.010E-02     8.522834E+01    8.52283E+01       0.0000E+00      ;%   Europa
                    1.07040E+06     0.000000E+00     0.000000E+00    0.00000E+00    0.00000E+00   7.1546E+00   0.000E+00
0.000E+00   0.000E+00    2.1000E-01     1.500E-03     1.717092E+02    1.71709E+02       0.0000E+00      ;%   Ganymede
                    1.88270E+06     0.000000E+00     0.000000E+00    0.00000E+00    0.00000E+00   1.6689E+01   0.000E+00
0.000E+00   0.000E+00    5.1000E-01     7.000E-03     4.005364E+02    4.00536E+02       0.0000E+00      ;%   Callisto
                    1.28000E+05     0.000000E+00     0.000000E+00    0.00000E+00    0.00000E+00   2.9478E-01   0.000E+00
0.000E+00   0.000E+00    6.0000E-02     2.000E-04     7.074696E+00    7.07470E+00       0.0000E+00      ;%   Metis
                    1.29000E+05     0.000000E+00     0.000000E+00    0.00000E+00    0.00000E+00   2.9826E-01   0.000E+00
0.000E+00   0.000E+00    3.0000E-02     1.500E-03     7.158240E+00    7.15824E+00       0.0000E+00      ;%   Adrastea
                    1.81400E+05     0.000000E+00     0.000000E+00    0.00000E+00    0.00000E+00   4.9818E-01   0.000E+00
0.000E+00   0.000E+00    4.0000E-01     3.000E-03     1.195630E+01    1.19563E+01       0.0000E+00      ;%   Amalthea
                    2.21900E+05     0.000000E+00     0.000000E+00    0.00000E+00    0.00000E+00   6.7450E-01   0.000E+00
0.000E+00   0.000E+00    8.0000E-01     1.800E-02     1.618800E+01    1.61880E+01       0.0000E+00      ;%   Thebe
                      1.43353E+12     1.075922E+04     1.074694E+04    1.35255E+12    1.51450E+12   3.7809E+02   9.690E+03
1.018E+04   9.090E+03    2.4850E+00     5.650E-02     1.065600E+01    1.06560E+01       2.6730E+01      ;% Saturn
                    1.85520E+05     0.000000E+00     0.000000E+00    0.00000E+00    0.00000E+00   9.4242E-01   0.000E+00
0.000E+00   0.000E+00    1.5300E+00     2.020E-02     2.216181E+01    2.21618E+01       0.0000E+00      ;%   Mimas
                    2.38020E+05     0.000000E+00     0.000000E+00    0.00000E+00    0.00000E+00   1.3702E+00   0.000E+00
0.000E+00   0.000E+00    0.0000E+00     4.500E-03     3.288523E+01    3.28852E+01       0.0000E+00      ;%   Enceladus
                    2.94660E+05     0.000000E+00     0.000000E+00    0.00000E+00    0.00000E+00   1.8878E+00   0.000E+00
0.000E+00   0.000E+00    1.8600E+00     0.000E+00     4.530725E+01    4.53073E+01       0.0000E+00      ;%   Tethys
                    3.77400E+05     0.000000E+00     0.000000E+00    0.00000E+00    0.00000E+00   2.7369E+00   0.000E+00
0.000E+00   0.000E+00    2.0000E-02     2.200E-03     6.568596E+01    6.56860E+01       0.0000E+00      ;%   Dione
                    5.27040E+05     0.000000E+00     0.000000E+00    0.00000E+00    0.00000E+00   4.5175E+00   0.000E+00
0.000E+00   0.000E+00    3.5000E-01     1.000E-03     1.084200E+02    1.08420E+02       0.0000E+00      ;%   Rhea
                    1.22183E+06     1.594500E+01     1.594500E+01    0.00000E+00    0.00000E+00   1.5945E+01   0.000E+00
0.000E+00   0.000E+00    3.3000E-01     2.920E-02     3.826901E+02    3.82690E+02       0.0000E+00      ;%   Titan
                    1.48110E+06     0.000000E+00     0.000000E+00    0.00000E+00    0.00000E+00   2.1277E+00   0.000E+00
0.000E+00   0.000E+00    4.3000E-01     1.042E-01     5.106386E+02    5.10639E+02       0.0000E+00      ;%   Hyperion
                    3.56130E+06     0.000000E+00     0.000000E+00    0.00000E+00    0.00000E+00   7.9330E+00   0.000E+00
0.000E+00   0.000E+00    1.4720E+01     2.830E-02     1.903924E+03    1.90392E+03       0.0000E+00      ;%   Iapetus
                                    2.87246E+12     3.068540E+04     3.058874E+04    2.74130E+12    3.00362E+12   3.6966E+02
6.810E+03    7.110E+03   6.490E+03     7.7200E-01    4.570E-02      -1.724000E+01    1.72400E+01       9.7770E+01      ;% Uranus
                                    4.49506E+12     6.018900E+04     5.979990E+04    4.44445E+12    4.54567E+12   3.6749E+02
5.430E+03    5.500E+03   5.370E+03     1.7690E+00    1.130E-02       1.611000E+01    1.61100E+01       2.8320E+01      ;% Neptune
                                    5.90638E+12     9.046500E+04     9.058800E+04    4.43682E+12    7.37593E+12   3.6673E+02
4.720E+03    6.100E+03   3.710E+03     1.7160E+01    2.488E-01      -1.532928E+02    1.53282E+02       1.2253E+02      };  % Pluto
```

```
    UnitLabelStr1 = {  'N/A'      'kg'    '10^10 m^3'    'm'    'm'    'm/s^2'    'deg C'    'Pascals'    'kg/m^3'    'N/A'    'm/s'
'kg^3/s^2'    'N/A'    'N/A'    'V(1,0)'    'W/m^2'    'deg K'    'kg/m^3'    'mGauss'    'I/MR^2'    'x10^-6'    'N/A'    };
    UnitLabelStr2 = {  'm'    'E. Days'    'E. Days'    'm'    'm'    'E. Days'    'm/s'    'm/s'    'm/s'    'deg'    'N/A'    'E.
Hours'    'E. Hours'    'deg'    };


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
mGauss        I/MR^2     x10^-6    SATELLITES   ]
    Planet_DB = {   'SUN'          4.3852E+30  4.98643E+18   2.2834E+09     2.2834E+09     9.012E+02     9940.73      1.015E+01
8.7899E+01   5.0000E-05    2.0263E+06    4.6866E+21   0.000E+00     0.000E+00  -26.74    0.0000E+00      9.941E+03    0.000E+00
0.00       5.900E-02  0.00000E+00      0          ;% Sun
                    'MERCURY'      7.2801E+23  2.14819E+11   8.0045E+06     8.0045E+06     1.214E+01     332.33      1.450E-13
3.3880E+02   0.0000E+00    1.4107E+04    7.7798E+14   1.190E-01     1.060E-01   -0.42    2.8931E+03     3.368E+02    0.000E+00
0.00       3.300E-01  6.00000E+01      0          ;% Mercury
                    'VENUS'        1.0734E+25  3.27872E+12   1.9855E+07     1.9855E+07     2.910E+01     854.33      1.334E+03
3.2731E+02   0.0000E+00    3.3990E+04    1.1474E+16   7.500E-01     6.500E-01   -4.40    8.2861E+02    -4.261E+01    4.020E+00
0.00       3.300E-01  4.45800E+00      0          ;% Venus
                                       'EARTH'        1.3171E+25  3.82532E+12   2.0926E+07     2.0926E+07     3.219E+01       59.00
1.470E+01   3.4429E+02   3.3500E-03    3.6745E+04     1.4076E+16   3.060E-01    3.670E-01   -3.86    4.3352E+02    -1.926E+00
2.377E-03      310.00       3.308E-01  1.08263E+03      1          ;% Earth
                                       'MOON'         1.6202E+23  7.75440E+10   5.7003E+06     5.6955E+06     5.315E+00       -9.67
4.351E-14   2.0913E+02   1.2000E-03    7.8084E+03     1.7304E+14   1.100E-01    1.200E-01   +0.21    4.3352E+02     3.443E+01
0.000E+00        0.00       3.940E-01  2.02700E+02      0          ;% Moon
                                       'MARS'         1.4150E+24  5.76265E+11   1.1145E+07     1.1073E+07     1.217E+01     -85.00
9.224E+01   2.4553E+02   6.4800E-03    1.6502E+03     1.5125E+15   2.500E-01    1.500E-01   -1.52    1.8678E+02    -8.149E-01
0.000E+00        0.00       3.660E-01  1.96045E+03      2          ;% Mars
                                       'JUPITER'      4.1857E+27  5.05452E+15   2.3455E+08     2.1934E+08     7.585E+01       58.73
1.450E+04   8.2779E+02   6.4870E-02    1.9521E+05     4.4740E+18   3.430E-01    5.200E-01   -9.40    1.6009E+01    -2.617E-02
0.000E+00      4280.00       2.540E-01  1.47360E+04     63          ;% Jupiter
                    'IO'           1.9692E+23  8.94136E+10   5.9764E+06     5.9764E+06     0.000E+00       0.00      0.000E+00
2.2037E+02   0.0000E+00    0.0000E+00    0.0000E+00   0.000E+00     6.200E-01     0.00    0.0000E+00      0.000E+00    0.000E+00
0.00       0.000E+00  0.00000E+00      0          ;%   Io
                    'EUROPA'       1.0582E+23  5.62453E+10   5.1207E+06     5.1207E+06     0.000E+00       0.00      0.000E+00
1.8791E+02   0.0000E+00    0.0000E+00    0.0000E+00   0.000E+00     6.800E-01     0.00    0.0000E+00      0.000E+00    0.000E+00
0.00       0.000E+00  0.00000E+00      0          ;%   Europa
                    'GANYMEDE'     3.2670E+23  2.69467E+11   8.6325E+06     8.6325E+06     0.000E+00       0.00      0.000E+00
1.2111E+02   0.0000E+00    0.0000E+00    0.0000E+00   0.000E+00     4.400E-01     0.00    0.0000E+00      0.000E+00    0.000E+00
0.00       0.000E+00  0.00000E+00      0          ;%   Ganymede
                    'CALLISTO'     2.3720E+23  2.07137E+11   7.9078E+06     7.9078E+06     0.000E+00       0.00      0.000E+00
1.1424E+02   0.0000E+00    0.0000E+00    0.0000E+00   0.000E+00     1.900E-01     0.00    0.0000E+00      0.000E+00    0.000E+00
0.00       0.000E+00  0.00000E+00      0          ;%   Callisto
                    'METIS'        2.2046E+17  1.18341E+06   6.5617E+04     6.5617E+04     0.000E+00       0.00      0.000E+00
0.0000E+00   0.0000E+00    0.0000E+00    0.0000E+00   0.000E+00     6.000E-02     0.00    0.0000E+00      0.000E+00    0.000E+00
0.00       0.000E+00  0.00000E+00      0          ;%   Metis
```

```
                    'ADRASTEA'    4.4092E+16  1.53843E+04   4.2651E+04     2.6247E+04     0.000E+00        0.00      0.000E+00
0.0000E+00    0.0000E+00     0.0000E+00     0.0000E+00    0.000E+00     1.000E-01     0.00     0.0000E+00      0.000E+00     0.000E+00
0.00       0.000E+00   0.00000E+00      0          ;%   Adrastea
                    'AMALTHEA'    1.6535E+19  9.47793E+06   4.2979E+05     2.1982E+05     0.000E+00        0.00      0.000E+00
1.9353E+02    0.0000E+00     0.0000E+00     0.0000E+00    0.000E+00     9.000E-02     0.00     0.0000E+00      0.000E+00     0.000E+00
0.00       0.000E+00   0.00000E+00      0          ;%   Amalthea
                    'THEBE'       1.7637E+18  2.01364E+06   1.8045E+05     1.4764E+05     0.000E+00        0.00      0.000E+00
0.0000E+00    0.0000E+00     0.0000E+00     0.0000E+00    0.000E+00     5.000E-02     0.00     0.0000E+00      0.000E+00     0.000E+00
0.00       0.000E+00   0.00000E+00      0          ;%   Thebe
                                          'SATURN'       1.2532E+27  2.92098E+15   1.9773E+08     1.7836E+08     2.940E+01     -218.47
1.450E+04    4.2888E+01    9.7960E-02     1.1647E+05     1.3395E+18    3.420E-01     4.700E-01    -8.88     4.7233E+00     -3.137E-02
0.000E+00       220.00       2.100E-01   1.62980E+04     56          ;% Saturn
                    'MIMAS'       8.3555E+19  1.15739E+08   6.8569E+05     6.2664E+05     0.000E+00        0.00      0.000E+00
7.1792E+01    0.0000E+00     0.0000E+00     0.0000E+00    0.000E+00     6.000E-01     0.00     0.0000E+00      0.000E+00     0.000E+00
0.00       0.000E+00   0.00000E+00      0          ;%   Mimas
                    'ENCELADUS'   2.3810E+20  2.29164E+08   8.3989E+05     8.0380E+05     0.000E+00        0.00      0.000E+00
1.0051E+02    0.0000E+00     0.0000E+00     0.0000E+00    0.000E+00     1.000E+00     0.00     0.0000E+00      0.000E+00     0.000E+00
0.00       0.000E+00   0.00000E+00      0          ;%   Enceladus
                    'TETHYS'      1.3625E+21  2.20206E+09   1.7585E+06     1.7257E+06     0.000E+00        0.00      0.000E+00
5.9931E+01    0.0000E+00     0.0000E+00     0.0000E+00    0.000E+00     8.000E-01     0.00     0.0000E+00      0.000E+00     0.000E+00
0.00       0.000E+00   0.00000E+00      0          ;%   Tethys
                    'DIONE'       2.4251E+21  2.59781E+09   1.8373E+06     1.8373E+06     0.000E+00        0.00      0.000E+00
9.1769E+02    0.0000E+00     0.0000E+00     0.0000E+00    0.000E+00     7.000E-01     0.00     0.0000E+00      0.000E+00     0.000E+00
0.00       0.000E+00   0.00000E+00      0          ;%   Dione
                    'RHEA'        5.0927E+21  6.59664E+09   2.5066E+06     2.5066E+06     0.000E+00        0.00      0.000E+00
7.6786E+02    0.0000E+00     0.0000E+00     0.0000E+00    0.000E+00     7.000E-01     0.00     0.0000E+00      0.000E+00     0.000E+00
0.00       0.000E+00   0.00000E+00      0          ;%   Rhea
                    'TITAN'       2.9663E+23  2.52566E+11   8.4481E+06     8.4481E+06     4.436E+00     -291.01      2.128E+01
1.1736E+02    0.0000E+00     8.6581E+03     0.0000E+00    2.200E-01     2.200E-01     7.90     4.7687E+00      0.000E+00     0.343E+00
0.00       0.000E+00   0.00000E+00      0          ;%   Titan
                    'HYPERION'    1.2125E+19  4.32933E+07   6.0695E+05     3.7073E+05     0.000E+00        0.00      0.000E+00
3.5584E+01    0.0000E+00     0.0000E+00     0.0000E+00    0.000E+00     3.000E-01     0.00     0.0000E+00      0.000E+00     0.000E+00
0.00       0.000E+00   0.00000E+00      0          ;%   Hyperion
                    'IAPETUS'     3.9904E+21  5.47540E+09   2.3556E+06     2.3556E+06     0.000E+00        0.00      0.000E+00
6.8046E+01    0.0000E+00     0.0000E+00     0.0000E+00    0.000E+00     5.000E-01     0.00     0.0000E+00      0.000E+00     0.000E+00
0.00       0.000E+00   0.00000E+00      0          ;%   Iapetus
                                          'URANUS'       1.9143E+26  2.41305E+14   8.3857E+07     8.1932E+07     2.851E+01     -322.87
1.450E+04    7.9284E+01    2.2930E-02     6.9881E+04     2.0461E+17    3.000E-01     5.100E-01    -7.19     1.1761E+00     -3.549E-02
0.000E+00       230.00       2.250E-01   3.34343E+03     27          ;% Uranus
                                          'NEPTUNE'      2.2582E+26  2.20858E+14   8.1249E+07     7.9858E+07     3.609E+01     -328.27
1.450E+04    1.0226E+01    1.7080E-02     7.7099E+04     2.4138E+17    2.900E-01     4.100E-01    -6.87     4.7867E-01     -3.758E-02
0.000E+00       140.00       0.000E+00   3.41100E+03     13          ;% Neptune
                                          'PLUTO'        2.7558E+22  2.52499E+11   3.7763E+06     3.9206E+06     2.657E+00     -355.63
4.351E-05    1.0925E+01    0.0000E+00     3.9370E+03     2.9311E+13    6.000E-01     6.000E-01    -1.00     2.8213E-01     -3.922E-02
0.000E+00         0.00       0.000E+00   0.00000E+00      3         };% Pluto

 %  Properties  [ SEMIMAJOR-Axis  SIDEREAL-Period  TROPICAL-Period   PERIHELION      APHELION      SYNODIC-Per   MEAN-VEL
MAX-VEL     MIN-VEL     ORB-INCLIN     ECCENTRICITY     ROTATION-Per   LENGTH-DAY    OBLIQUITY to Orbit    ]
 %    Units              ft              E. Days          E. Days          ft             ft            E. Days       ft/s
ft/s        ft/s          deg             N/A            E. Hours       E. Hours         deg             ]
    Planet_DB2 = {  0.00000E+00     0.000000E+00     0.000000E+00    0.00000E+00    0.00000E+00   0.0000E+00   6.365E+04
0.000E+00    0.000E+00     0.0000E+00     0.000E+00      0.000000E+00    0.00000E+00      0.0000E+00      ;% Sun
                  1.89993E+11     8.796900E+01     8.796800E+01    1.50918E+11    2.26115E+11   1.1588E+02   1.571E+05
1.935E+05    1.275E+05     7.0000E+00     2.056E-01      1.407600E+03    4.22260E+03       1.0000E-02      ;% Mercury
```

```
                3.55019E+11     2.247010E+02     2.246950E+02    3.52623E+11    3.57414E+11   5.8392E+02   1.149E+05
1.157E+05   1.141E+05   3.3900E+00    6.700E-02     -5.832500E+03    2.80200E+03       1.7736E+02     ;% Venus
                                4.90812E+11     3.652560E+02     3.652420E+02    4.82577E+11    4.99014E+11   0.0000E+00
9.770E+04    9.938E+04   9.610E+04    0.0000E+00    1.670E-02       2.393450E+01    2.40000E+01       2.3450E+01     ;% Earth
                          1.26115E+09     0.000000E+00     0.000000E+00    1.19193E+09    1.33038E+09   2.9530E+01
3.356E+03    3.530E+03   3.163E+03    5.1450E+00    5.490E-02       6.557280E+02    0.00000E+00       6.6800E+00     ;% Moon
                                7.47767E+11     6.869800E+02     6.869730E+02    6.77885E+11    8.17681E+11   7.7994E+02
7.917E+04    8.694E+04   7.208E+04    1.8500E+00    9.350E-02       2.462290E+01    2.46597E+01       2.5190E+01     ;% Mars
                                2.55436E+12     4.332589E+03     4.330595E+03    2.42952E+12    2.67919E+12   3.9888E+02
4.288E+04    4.501E+04   4.081E+04    1.3040E+00    4.890E-02       9.925000E+00    9.92590E+00       3.1300E+00     ;% Jupiter
                1.38320E+06     0.000000E+00     0.000000E+00    0.00000E+00    0.00000E+00   1.7691E+00   0.000E+00
0.000E+00   0.000E+00   4.0000E-02    4.000E-03     4.245931E+01    4.24593E+01       0.0000E+00     ;%   Io
                6.70900E+05     0.000000E+00     0.000000E+00    0.00000E+00    0.00000E+00   3.5512E+00   0.000E+00
0.000E+00   0.000E+00   4.7000E-01    1.010E-02     8.522834E+01    8.52283E+01       0.0000E+00     ;%   Europa
                2.20111E+06     0.000000E+00     0.000000E+00    0.00000E+00    0.00000E+00   7.1546E+00   0.000E+00
0.000E+00   0.000E+00   2.1000E-01    1.500E-03     1.717092E+02    1.71709E+02       0.0000E+00     ;%   Ganymede
                6.17682E+06     0.000000E+00     0.000000E+00    0.00000E+00    0.00000E+00   1.6689E+01   0.000E+00
0.000E+00   0.000E+00   5.1000E-01    7.000E-03     4.005364E+02    4.00536E+02       0.0000E+00     ;%   Callisto
                4.19946E+05     0.000000E+00     0.000000E+00    0.00000E+00    0.00000E+00   2.9478E-01   0.000E+00
0.000E+00   0.000E+00   6.0000E-02    2.000E-04     7.074696E+00    7.07470E+00       0.0000E+00     ;%   Metis
                4.23227E+05     0.000000E+00     0.000000E+00    0.00000E+00    0.00000E+00   2.9826E-01   0.000E+00
0.000E+00   0.000E+00   3.0000E-02    1.500E-03     7.158240E+00    7.15824E+00       0.0000E+00     ;%   Adrastea
                5.95143E+05     0.000000E+00     0.000000E+00    0.00000E+00    0.00000E+00   4.9818E-01   0.000E+00
0.000E+00   0.000E+00   4.0000E-01    3.000E-03     1.195630E+01    1.19563E+01       0.0000E+00     ;%   Amalthea
                7.28016E+05     0.000000E+00     0.000000E+00    0.00000E+00    0.00000E+00   6.7450E-01   0.000E+00
0.000E+00   0.000E+00   8.0000E-01    1.800E-02     1.618800E+01    1.61880E+01       0.0000E+00     ;%   Thebe
                  4.70317E+12     1.075922E+04     1.074694E+04    4.43749E+12    4.96882E+12   3.7809E+02   3.179E+04
3.340E+04    2.982E+04   2.4850E+00    5.650E-02     1.065600E+01    1.06560E+01       2.6730E+01     ;% Saturn
                6.08660E+05     0.000000E+00     0.000000E+00    0.00000E+00    0.00000E+00   9.4242E-01   0.000E+00
0.000E+00   0.000E+00   1.5300E+00    2.020E-02     2.216181E+01    2.21618E+01       0.0000E+00     ;%   Mimas
                7.80903E+05     0.000000E+00     0.000000E+00    0.00000E+00    0.00000E+00   1.3702E+00   0.000E+00
0.000E+00   0.000E+00   0.0000E+00    4.500E-03     3.288523E+01    3.28852E+01       0.0000E+00     ;%   Enceladus
                9.66729E+05     0.000000E+00     0.000000E+00    0.00000E+00    0.00000E+00   1.8878E+00   0.000E+00
0.000E+00   0.000E+00   1.8600E+00    0.000E+00     4.530725E+01    4.53073E+01       0.0000E+00     ;%   Tethys
                1.23819E+06     0.000000E+00     0.000000E+00    0.00000E+00    0.00000E+00   2.7369E+00   0.000E+00
0.000E+00   0.000E+00   2.0000E-02    2.200E-03     6.568596E+01    6.56860E+01       0.0000E+00     ;%   Dione
                1.72913E+06     0.000000E+00     0.000000E+00    0.00000E+00    0.00000E+00   4.5175E+00   0.000E+00
0.000E+00   0.000E+00   3.5000E-01    1.000E-03     1.084200E+02    1.08420E+02       0.0000E+00     ;%   Rhea
                1.22183E+06     1.594500E+01     1.594500E+01    0.00000E+00    0.00000E+00   1.5945E+01   0.000E+00
0.000E+00   0.000E+00   3.3000E-01    2.920E-02     3.826901E+02    3.82690E+02       0.0000E+00     ;%   Titan
                4.00862E+06     0.000000E+00     0.000000E+00    0.00000E+00    0.00000E+00   2.1277E+00   0.000E+00
0.000E+00   0.000E+00   4.3000E-01    1.042E-01     5.106386E+02    5.10639E+02       0.0000E+00     ;%   Hyperion
                1.16840E+07     0.000000E+00     0.000000E+00    0.00000E+00    0.00000E+00   7.9330E+00   0.000E+00
0.000E+00   0.000E+00   1.4720E+01    2.830E-02     1.903924E+03    1.90392E+03       0.0000E+00     ;%   Iapetus
                                9.42405E+12     3.068540E+04     3.058874E+04    8.99374E+12    9.85437E+12   3.6966E+02
2.234E+04    2.333E+04   2.129E+04    7.7200E-01    4.570E-02      -1.724000E+01    1.72400E+01       9.7770E+01     ;% Uranus
                                1.47475E+13     6.018900E+04     5.979990E+04    1.45815E+13    1.49136E+13   3.6749E+02
1.781E+04    1.804E+04   1.762E+04    1.7690E+00    1.130E-02       1.611000E+01    1.61100E+01       2.8320E+01     ;% Neptune
                                1.93778E+13     9.046500E+04     9.058800E+04    1.45565E+13    2.41992E+13   3.6673E+02
1.549E+04    2.000E+04   1.217E+04    1.7160E+01    2.488E-01      -1.532928E+02    1.53282E+02       1.2253E+02     };  % Pluto

    UnitLabelStr1 = {  'N/A'   'lbm'    '10^10 ft^3'    'ft'    'ft'    'ft/s^2'    'deg F'    'psi'    'lbm/ft^3'    'N/A'    'ft/s'
'lbm^3/s^2'    'N/A'    'N/A'    'V(1,0)'    'BTUs/h*ft^2'    'deg F'    'lbm/ft^3'    'mGauss'    'I/MR^2'    'x10^-6'    'N/A'    };
```

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

end

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

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

%  Order of Decreasing Compositon
%                    { Element    %Concentration    Element  %Concentration   Element   %Concentration . . .  }
Planet_AtmosphereDB = {     'H'        '90.965'       'He'        '8.889'       'N/A'      '0.0'      'N/A'  '0.0'      'N/A'
'0.0'     'N/A'   '0.0'     ;  % Sun
                            'O2'       '42'           'Na'        '29'          'H2'       '22'       'He'   '6'        'K'
'0.5'     'N/A'   '0.0'     ;  % Mercury
                            'CO2'      '96.5'         'N2'        '3.5'         'N/A'      '0.0'      'N/A'  '0.0'      'N/A'
'0.0'     'N/A'   '0.0'     ;  % Venus
                            'N2'       '78.084'       'O2'        '20.946'      'H2O'      '1.0'      'N/A'  '0.0'      'N/A'
'0.0'     'N/A'   '0.0'     ;  % Earth
                            'N/A'      '0.0'          'N/A'       '0.0'         'N/A'      '0.0'      'N/A'  '0.0'      'N/A'
'0.0'     'N/A'   '0.0'     ;  % Moon
                            'CO2'      '95.32'        'N2'        '2.7'         'Are'       '1.6'      'O2'  '0.13'      'CO'
'0.08'    'N/A'   '0.0'     ;  % Mars
                            'H2'       '89.8'         'He'        '10.2'        'N/A'      '0.0'      'N/A'  '0.0'      'N/A'
'0.0'     'N/A'   '0.0'     ;  % Jupiter
                            'N/A'      '0.0'          'N/A'       '0.0'         'N/A'      '0.0'      'N/A'  '0.0'      'N/A'
'0.0'     'N/A'   '0.0'     ;  %   Io
                            'N/A'      '0.0'          'N/A'       '0.0'         'N/A'      '0.0'      'N/A'  '0.0'      'N/A'
'0.0'     'N/A'   '0.0'     ;  %   Europa
                            'N/A'      '0.0'          'N/A'       '0.0'         'N/A'      '0.0'      'N/A'  '0.0'      'N/A'
'0.0'     'N/A'   '0.0'     ;  %   Ganymede
                            'N/A'      '0.0'          'N/A'       '0.0'         'N/A'      '0.0'      'N/A'  '0.0'      'N/A'
'0.0'     'N/A'   '0.0'     ;  %   Callisto
                            'N/A'      '0.0'          'N/A'       '0.0'         'N/A'      '0.0'      'N/A'  '0.0'      'N/A'
'0.0'     'N/A'   '0.0'     ;  %   Metis
                            'N/A'      '0.0'          'N/A'       '0.0'         'N/A'      '0.0'      'N/A'  '0.0'      'N/A'
'0.0'     'N/A'   '0.0'     ;  %   Adrastea
                            'N/A'      '0.0'          'N/A'       '0.0'         'N/A'      '0.0'      'N/A'  '0.0'      'N/A'
'0.0'     'N/A'   '0.0'     ;  %   Amalthea
                            'N/A'      '0.0'          'N/A'       '0.0'         'N/A'      '0.0'      'N/A'  '0.0'      'N/A'
'0.0'     'N/A'   '0.0'     ;  %   Thebe
                            'H2'       '96.3'         'He'        '3.25'        'N/A'      '0.0'      'N/A'  '0.0'      'N/A'
'0.0'     'N/A'   '0.0'     ;  % Saturn
                            'N/A'      '0.0'          'N/A'       '0.0'         'N/A'      '0.0'      'N/A'  '0.0'      'N/A'
'0.0'     'N/A'   '0.0'     ;  %   Mimas
                            'N/A'      '0.0'          'N/A'       '0.0'         'N/A'      '0.0'      'N/A'  '0.0'      'N/A'
'0.0'     'N/A'   '0.0'     ;  %   Enceladus
                            'N/A'      '0.0'          'N/A'       '0.0'         'N/A'      '0.0'      'N/A'  '0.0'      'N/A'
'0.0'     'N/A'   '0.0'     ;  %   Tethys
                            'N/A'      '0.0'          'N/A'       '0.0'         'N/A'      '0.0'      'N/A'  '0.0'      'N/A'
'0.0'     'N/A'   '0.0'     ;  %   Dione
                            'N/A'      '0.0'          'N/A'       '0.0'         'N/A'      '0.0'      'N/A'  '0.0'      'N/A'
'0.0'     'N/A'   '0.0'     ;  %   Rhea
```

```
                        'N2'        '98.4'          'CH4'       '1.6'          'N/A'       '0.0'       'N/A'   '0.0'       'N/A'
'0.0'     'N/A'   '0.0'     ;  %   Titan
                        'N/A'       '0.0'           'N/A'       '0.0'          'N/A'       '0.0'       'N/A'   '0.0'       'N/A'
'0.0'     'N/A'   '0.0'     ;  %   Hyperion
                        'N/A'       '0.0'           'N/A'       '0.0'          'N/A'       '0.0'       'N/A'   '0.0'       'N/A'
'0.0'     'N/A'   '0.0'     ;  %   Iapetus
                        'H2'        '82.5'          'He'        '15.2'         'CH4'       '2.3'       'N/A'   '0.0'       'N/A'
'0.0'     'N/A'   '0.0'     ;  % Uranus
                        'H2'        '80.0'          'He'        '19.0'         'CH4'       '1.5'       'N/A'   '0.0'       'N/A'
'0.0'     'N/A'   '0.0'     ;  % Neptune
                        'CH4'       '0.0'           'N2'        '0.0'          'N/A'       '0.0'       'N/A'   '0.0'       'N/A'
'0.0'     'N/A'   '0.0'     }  ;  % Pluto

%                       Trace elements
%                       Element   Conc  Conc Unit    Element   Conc  Conc Unit      ...
Planet_AtmosphereDB2 = {  'O'     '774'   'ppm'         'C'     '330'   'ppm'        'Ne'    '112'   'ppm'       'N'     '102'   'ppm'
'Fe'    '43'    'ppm'       'Mg'    '35'    'ppm'     'S'     '15'    'ppm'      'N/A'  '0'     'ppm'     'N/A'  '0'     'ppm'   ; % Sun
                            'Are'    '774'    'ppm'          'CO2'   '330'    'ppm'         'H2O'   '112'    'ppm'       'N2'    '102'    'ppm'
'Xe'    '43'    'ppm'       'Kr'    '35'    'ppm'     'Ne'    '15'    'ppm'      'N/A'  '0'     'ppm'     'N/A'  '0'     'ppm'   ; % Mercury
                            'SO2'   '150'   'ppm'         'Are'    '70'    'ppm'        'H2O'   '20'    'ppm'       'CO'    '17'    'ppm'
'He'    '12'    'ppm'       'Ne'    '7'     'ppm'     'N/A'  '0'     'ppm'      'N/A'  '0'     'ppm'     'N/A'  '0'     'ppm'   ; % Venus
                            'Are'    '9340'   'ppm'          'CO2'   '380'    'ppm'         'Ne'    '18.18' 'ppm'       'He'    '5.24'   'ppm'
'CH4'   '1.7'   'ppm'       'Kr'    '1.14'  'ppm'     'H2'    '.55'   'ppm'      'N/A'  '0'     'ppm'     'N/A'  '0'     'ppm'   ; % Earth
                            'He4'   '40000' 'ppccm'       'Ne20'  '40000' 'ppccm'      'H2'    '35000' 'ppccm'     'Ar40'  '30000' 'ppccm'
'Ne22'  '5000'  'ppccm'     'Ar36'  '2000'  'ppccm'   'CH4'   '1000'  'ppccm'    'NH3'  '1000'  'ppccm'    'C02'  '1000'  'ppccm' ; % Moon
                            'H2O'   '210'    'ppm'          'NO'    '100'   'ppm'        'Ne'    '2.5'   'ppm'       'HDO'   '0.85'   'ppm'
'Kr'    '0.3'   'ppm'       'Xe'    '0.08'  'ppm'     'N/A'  '0'     'ppm'      'N/A'  '0'     'ppm'     'N/A'  '0'     'ppm'   ; % Mars
                            'CH4'   '3000'  'ppm'         'NH3'   '260'   'ppm'        'HD'    '28'    'ppm'       'C2H6'  '5.8'    'ppm'
'H2O'   '4'     'ppm'       'N/A'   '0'     'ppm'     'N/A'  '0'     'ppm'      'N/A'  '0'     'ppm'     'N/A'  '0'     'ppm'   ; % Jupiter
                            'N/A'   '0'     'ppm'         'N/A'   '0'     'ppm'        'N/A'   '0'     'ppm'       'N/A'   '0'     'ppm'
'N/A'   '0'     'ppm'       'N/A'   '0'     'ppm'     'N/A'  '0'     'ppm'      'N/A'  '0'     'ppm'     'N/A'  '0'     'ppm'   ; %   Io
                            'N/A'   '0'     'ppm'         'N/A'   '0'     'ppm'        'N/A'   '0'     'ppm'       'N/A'   '0'     'ppm'
'N/A'   '0'     'ppm'       'N/A'   '0'     'ppm'     'N/A'  '0'     'ppm'      'N/A'  '0'     'ppm'     'N/A'  '0'     'ppm'   ; %   Europa
                            'N/A'   '0'     'ppm'         'N/A'   '0'     'ppm'        'N/A'   '0'     'ppm'       'N/A'   '0'     'ppm'
'N/A'   '0'     'ppm'       'N/A'   '0'     'ppm'     'N/A'  '0'     'ppm'      'N/A'  '0'     'ppm'     'N/A'  '0'     'ppm'   ; %
Ganymede
                            'N/A'   '0'     'ppm'         'N/A'   '0'     'ppm'        'N/A'   '0'     'ppm'       'N/A'   '0'     'ppm'
'N/A'   '0'     'ppm'       'N/A'   '0'     'ppm'     'N/A'  '0'     'ppm'      'N/A'  '0'     'ppm'     'N/A'  '0'     'ppm'   ; %
Callisto
                            'N/A'   '0'     'ppm'         'N/A'   '0'     'ppm'        'N/A'   '0'     'ppm'       'N/A'   '0'     'ppm'
'N/A'   '0'     'ppm'       'N/A'   '0'     'ppm'     'N/A'  '0'     'ppm'      'N/A'  '0'     'ppm'     'N/A'  '0'     'ppm'   ; %   Metis
                            'N/A'   '0'     'ppm'         'N/A'   '0'     'ppm'        'N/A'   '0'     'ppm'       'N/A'   '0'     'ppm'
'N/A'   '0'     'ppm'       'N/A'   '0'     'ppm'     'N/A'  '0'     'ppm'      'N/A'  '0'     'ppm'     'N/A'  '0'     'ppm'   ; %
Adrastea
                            'N/A'   '0'     'ppm'         'N/A'   '0'     'ppm'        'N/A'   '0'     'ppm'       'N/A'   '0'     'ppm'
'N/A'   '0'     'ppm'       'N/A'   '0'     'ppm'     'N/A'  '0'     'ppm'      'N/A'  '0'     'ppm'     'N/A'  '0'     'ppm'   ; %
Amalthea
                            'N/A'   '0'     'ppm'         'N/A'   '0'     'ppm'        'N/A'   '0'     'ppm'       'N/A'   '0'     'ppm'
'N/A'   '0'     'ppm'       'N/A'   '0'     'ppm'     'N/A'  '0'     'ppm'      'N/A'  '0'     'ppm'     'N/A'  '0'     'ppm'   ; %   Thebe
                            'CH4'   '4500'  'ppm'         'NH3'   '125'   'ppm'        'HD'    '110'   'ppm'       'C2H6'  '7'     'ppm'
'N/A'   '0'     'ppm'       'N/A'   '0'     'ppm'     'N/A'  '0'     'ppm'      'N/A'  '0'     'ppm'     'N/A'  '0'     'ppm'   ; % Saturn

                            'N/A'   '0'     'ppm'         'N/A'   '0'     'ppm'        'N/A'   '0'     'ppm'       'N/A'   '0'     'ppm'
'N/A'   '0'     'ppm'       'N/A'   '0'     'ppm'     'N/A'  '0'     'ppm'      'N/A'  '0'     'ppm'     'N/A'  '0'     'ppm'   ; %   Mimas
```

```
                        'N/A'  '0'     'ppm'       'N/A'  '0'     'ppm'      'N/A'  '0'     'ppm'      'N/A'  '0'     'ppm'
'N/A'  '0'    'ppm'     'N/A'  '0'    'ppm'    'N/A'  '0'    'ppm'    'N/A' '0'    'ppm'    'N/A' '0'    'ppm'   ; %
Enceladus
                        'N/A'  '0'     'ppm'       'N/A'  '0'     'ppm'      'N/A'  '0'     'ppm'      'N/A'  '0'     'ppm'
'N/A'  '0'    'ppm'     'N/A'  '0'    'ppm'    'N/A'  '0'    'ppm'    'N/A' '0'    'ppm'    'N/A' '0'    'ppm'   ; %   Tethys
                        'N/A'  '0'     'ppm'       'N/A'  '0'     'ppm'      'N/A'  '0'     'ppm'      'N/A'  '0'     'ppm'
'N/A'  '0'    'ppm'     'N/A'  '0'    'ppm'    'N/A'  '0'    'ppm'    'N/A' '0'    'ppm'    'N/A' '0'    'ppm'   ; %   Dione
                        'N/A'  '0'     'ppm'       'N/A'  '0'     'ppm'      'N/A'  '0'     'ppm'      'N/A'  '0'     'ppm'
'N/A'  '0'    'ppm'     'N/A'  '0'    'ppm'    'N/A'  '0'    'ppm'    'N/A' '0'    'ppm'    'N/A' '0'    'ppm'   ; %   Rhea

                        'N/A'  '0'     'ppm'       'N/A'  '0'     'ppm'      'N/A'  '0'     'ppm'      'N/A'  '0'     'ppm'
'N/A'  '0'    'ppm'     'N/A'  '0'    'ppm'    'N/A'  '0'    'ppm'    'N/A' '0'    'ppm'    'N/A' '0'    'ppm'   ; %   Titan
                        'N/A'  '0'     'ppm'       'N/A'  '0'     'ppm'      'N/A'  '0'     'ppm'      'N/A'  '0'     'ppm'
'N/A'  '0'    'ppm'     'N/A'  '0'    'ppm'    'N/A'  '0'    'ppm'    'N/A' '0'    'ppm'    'N/A' '0'    'ppm'   ; %
Hyperion
                        'N/A'  '0'     'ppm'       'N/A'  '0'     'ppm'      'N/A'  '0'     'ppm'      'N/A'  '0'     'ppm'
'N/A'  '0'    'ppm'     'N/A'  '0'    'ppm'    'N/A'  '0'    'ppm'    'N/A' '0'    'ppm'    'N/A' '0'    'ppm'   ; %   Iapetus
                        'HD'   '148'   'ppm'       'N/A'  '0'     'ppm'      'N/A'  '0'     'ppm'      'N/A'  '0'     'ppm'
'N/A'  '0'    'ppm'     'N/A'  '0'    'ppm'    'N/A'  '0'    'ppm'    'N/A' '0'    'ppm'    'N/A' '0'    'ppm'   ; % Uranus

                        'HD'   '192'   'ppm'       'C2H6' '1.5'   'ppm'      'N/A'  '0'     'ppm'      'N/A'  '0'     'ppm'
'N/A'  '0'    'ppm'     'N/A'  '0'    'ppm'    'N/A'  '0'    'ppm'    'N/A' '0'    'ppm'    'N/A' '0'    'ppm'   ; % Neptune
                        'N/A'  '0'     'ppm'       'N/A'  '0'     'ppm'      'N/A'  '0'     'ppm'      'N/A'  '0'     'ppm'
'N/A'  '0'    'ppm'     'N/A'  '0'    'ppm'    'N/A'  '0'    'ppm'    'N/A' '0'    'ppm'    'N/A' '0'    'ppm'   }; % Pluto
```



```
%  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('\n\n  ***  WARNING:  Planetary Atmospheric Density Undefined.  ***');
    fprintf('\n                Extrapolating Value From Ideal Gas Relation!\n\n');

    %  Loads Gas Constant Values Based on Unit System
    %  REF:  http://www.engineeringtoolbox.com/individual-universal-gas-constant-d_588.html
    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 'H2'    ;  R = 4124.0;     % J/kg*K
            case 'CH4'   ;  R =  518.3;     % J/kg*K
            case 'N2'    ;  R =  296.8;     % J/kg*K
            case 'O2'    ;  R =  258.8;     % J/kg*K
            case 'SO2'   ;  R =  130.0;     % J/kg*K
            case 'H2O'   ;  R =  461.5;     % J/kg*K
            otherwise
                fprintf('\n*** 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
        %  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_C2K)) ;

    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 =  1130.0;     % ft*lb/slug*R
            case 'He'   ;  R = 12420.0;     % ft*lb/slug*R
            case 'H2'   ;  R = 24660.0;     % ft*lb/slug*R
            case 'CH4'  ;  R =  3099.0;     % ft*lb/slug*R
            case 'N2'   ;  R =  1775.0;     % ft*lb/slug*R
            case 'O2'   ;  R =  1554.0;     % ft*lb/slug*R
            case 'SO2'  ;  R =     0.0;     % ft*lb/slug*R - not Defined
            case 'H2O'  ;  R =  2760.0;     % ft*lb/slug*R
            otherwise
                fprintf('\n*** 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
        %  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

end
%  Ends Definition of Planetary Atmospheric Density Section


%*************************************************************************************************************************
%
%   ***   PRINT SUMMARY Output Section for 'Print' = 'Y'

if strcmpi(Print,'Y') == 1

    fprintf('\n\n   ***   P L A N E T A R Y   D A T A B A S E   S U M M A R Y   R E S U L T S   ***');
    fprintf('\n\n***   I N P U T S   ***');
    fprintf('\n  Planetary Body Name:   %s', Planet_DB{LOC,1});
    fprintf('\n  Unit System:           %s', UNITS);

    fprintf('\n\n*************************************************************************\n\n');

    fprintf('***  PLANETARY BULK PROPERTIES  ***');
    fprintf('\n  Planetary Mass :                    %14.5e  %s', Planet_DB{LOC,2},  MassUnit );
    fprintf('\n  Planetary Volume (10^10) :          %14.5e  %s', Planet_DB{LOC,3},  VolUnit );
```

```
fprintf('\n  Planetary Equatorial Radius :          %14.5e  %s', Planet_DB{LOC,4},  DistUnit );
fprintf('\n  Planetary Polar Radius :               %14.5e  %s', Planet_DB{LOC,5},  DistUnit );
fprintf('\n  Planetary Gravity :                    %14.5e  %s', Planet_DB{LOC,6},  GravUnit );
fprintf('\n  Planetary Density :                    %14.5e  %s', Planet_DB{LOC,9},  DensityUnit );
fprintf('\n  Planetary Ellipticity :                %14.5e    ', Planet_DB{LOC,10} );
fprintf('\n  Planetary Escape Velocity :            %14.5e  %s', Planet_DB{LOC,11}, VelUnit );
fprintf('\n  Planetary GM :                         %14.5e  %s', Planet_DB{LOC,12}, GMUnit );
fprintf('\n  Planetary Bond Albedo :                %14.5e    ', Planet_DB{LOC,13} );
fprintf('\n  Planetary Visual Geometric Albedo :    %14.5e    ', Planet_DB{LOC,14} );
fprintf('\n  Planetary Visual Magnitude :           %14.5e    ', Planet_DB{LOC,15} );
fprintf('\n  Planetary Solar Irradiance :           %14.5e  %s', Planet_DB{LOC,16}, SolIradUnit );
fprintf('\n  Planetary Black Body Temperature :     %14.5e  %s', Planet_DB{LOC,17}, TempUnit );
fprintf('\n  Planetary Magnetic Field Strength:     %14.5e  %s', Planet_DB{LOC,19}, UnitLabelStr1{19} );
fprintf('\n  Planetary Moment of Inertia :          %14.5e    ', Planet_DB{LOC,20} );
fprintf('\n  Planetary J2 Parameter :               %14.5e    ', Planet_DB{LOC,21} );
fprintf('\n  Planetary Satellites :                 %14d      ', Planet_DB{LOC,22} );


fprintf('\n\n\n***  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  Sidereal Rotation Period :             %14.5e  Earth Hours', 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('\n\n*** ATMOSPHERIC PPROPERTIES  ***');
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\n   ATMOSPHERIC COMPOSITION   ');
fprintf('\n Major Elements ');
fprintf('\n-----------------------------');
fprintf('\n  Element   %% Composition   ');
for i = 1:2:length(PlanAtm_Prop)-1
    if strcmpi( PlanAtm_Prop{i}, 'N/A' ) ~= 1
        fprintf('\n  %6s        %7.3f %%', PlanAtm_Prop{i}, str2double(PlanAtm_Prop{i+1}) );
    end
end

 fprintf('\n\n Trace Elements ');
 fprintf('\n-----------------------------');
 fprintf('\n  Element      Amount     Concentration   ');
for i = 1:3:length(PlanAtm_MEProp)-2
```

```
        if strcmpi( PlanAtm_MEProp{i}, 'N/A' ) ~= 1
            fprintf('\n  %6s    %9.3f      %6s ', PlanAtm_MEProp{i}, str2double(PlanAtm_MEProp{i+1}), PlanAtm_MEProp{i+2} );
        end
    end
end

Planet_Properties = { Planet_DB{LOC,:} Planet_DB2{LOC,:} };
UnitLabelStr = [ UnitLabelStr1 UnitLabelStr2 ];

save('Planet_Data.mat', 'Planet_Properties', 'UnitLabelStr', 'PlanAtm_Prop', 'PlanAtm_MEProp' );



%************************************************************************
%***     Writes a text file summary of the Planetary Properties

    Fout = fopen('PlanetDB_Summary.txt','w+');

    fprintf(Fout, '\n\n   ***   P L A N E T A R Y   D A T A B A S E   S U M M A R Y   R E S U L T S   ***');
    fprintf(Fout, '\n\n***   I N P U T S   ***');
    fprintf(Fout, '\n  Planetary Body Name:   %s', Planet_DB{LOC,1});
    fprintf(Fout, '\n  Unit System:           %s', UNITS);

    fprintf(Fout, '\n\n*********************************************************************************\n\n');

    fprintf(Fout, '***  PLANETARY BULK PROPERTIES  ***');
    fprintf(Fout, '\n  Planetary Mass :                      %14.5e  %s', Planet_DB{LOC,2},  MassUnit );
    fprintf(Fout, '\n  Planetary Volume (10^10) :            %14.5e  %s', Planet_DB{LOC,3},  VolUnit );
    fprintf(Fout, '\n  Planetary Equatorial Radius :         %14.5e  %s', Planet_DB{LOC,4},  DistUnit );
    fprintf(Fout, '\n  Planetary Polar Radius :              %14.5e  %s', Planet_DB{LOC,5},  DistUnit );
    fprintf(Fout, '\n  Planetary Gravity :                   %14.5e  %s', Planet_DB{LOC,6},  GravUnit );
    fprintf(Fout, '\n  Planetary Density :                   %14.5e  %s', Planet_DB{LOC,9},  DensityUnit );
    fprintf(Fout, '\n  Planetary Ellipticity :               %14.5e    ', Planet_DB{LOC,10} );
    fprintf(Fout, '\n  Planetary Escape Velocity :           %14.5e  %s', Planet_DB{LOC,11}, VelUnit );
    fprintf(Fout, '\n  Planetary GM :                        %14.5e  %s', Planet_DB{LOC,12}, GMUnit );
    fprintf(Fout, '\n  Planetary Bond Albedo :               %14.5e    ', Planet_DB{LOC,13} );
    fprintf(Fout, '\n  Planetary Visual Geometric Albedo :   %14.5e    ', Planet_DB{LOC,14} );
    fprintf(Fout, '\n  Planetary Visual Magnitude :          %14.5e    ', Planet_DB{LOC,15} );
    fprintf(Fout, '\n  Planetary Solar Irradiance :          %14.5e  %s', Planet_DB{LOC,16}, SolIradUnit );
    fprintf(Fout, '\n  Planetary Black Body Temperature :    %14.5e  %s', Planet_DB{LOC,17}, TempUnit );
    fprintf(Fout, '\n  Planetary Magnetic Field Strength:    %14.5e  %s', Planet_DB{LOC,19}, UnitLabelStr1{19} );
    fprintf(Fout, '\n  Planetary Moment of Inertia :         %14.5e    ', Planet_DB{LOC,20} );
    fprintf(Fout, '\n  Planetary J2 Parameter :              %14.5e    ', Planet_DB{LOC,21} );
    fprintf(Fout, '\n  Planetary Satellites :                %14d    ', Planet_DB{LOC,22} );


    fprintf(Fout, '\n\n\n***  PLANETARY ORBITAL PARAMETERS  ***');
    fprintf(Fout, '\n  Semi-Major Axis :                     %14.5e  %s', Planet_DB2{LOC,1},  DistUnit );
    fprintf(Fout, '\n  Sidereal Orbit Period :               %14.5e  Earth Days', Planet_DB2{LOC,2} );
    fprintf(Fout, '\n  Tropical Orbit Period :               %14.5e  Earth Days', Planet_DB2{LOC,3} );
    fprintf(Fout, '\n  Perihelion Distance :                 %14.5e  %s', Planet_DB2{LOC,4}, DistUnit );
    fprintf(Fout, '\n  Aphelion Distance :                   %14.5e  %s', Planet_DB2{LOC,5}, DistUnit );
    fprintf(Fout, '\n  Synodic Period :                      %14.5e  Earth Days', Planet_DB2{LOC,6} );
```

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

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


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

     fprintf(Fout, '\n\n Trace Elements ');
     fprintf(Fout, '\n-----------------------------');
     fprintf(Fout, '\n  Element      Amount     Concentration   ');
    for i = 1:3:length(PlanAtm_MEProp)-2
        if strcmpi( PlanAtm_MEProp{i}, 'N/A' ) ~= 1
            fprintf(Fout, '\n  %6s    %9.3f      %6s ', PlanAtm_MEProp{i}, str2double(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');

PLTEMP = Planet_Properties{7};

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

%  AC Field Sensor Common Properties
% Properties          [    SENSOR       AXIS       FIELD RANGE       FREQUENCY RANGE      ACCURACY     ACCURACY     TIME INTERVAL
TOLERANCE  VOLTAGE   TYPE   ]
% Units               [       #           #       Low    mG   High     Low   Hz   High        %          Digits          sec
mV/mG       V      N/A    ]
AC_FIELD_SENSOR_DB1 = {   'Bell-4080'     3       0.1         511      25         1000        5           2             0.40
0        9      'DC'    ;
                           'MSI-95'       1       0.1       50000      25         3000        3           1             0.40
1.00       9      'DC'    ;
                           'Bell-4180'    3       0.1         599      30         2000        2           1             0.50
10.0       2      'DC'    ;
                           'Bell-4190'    3       0.1        1999      30         2000        1           1             0.50
10.0       2      'DC'    ;
                          'Bell-5170ULR'  3       0.1        1000       0        20000        2           0             0.25
0        5      'DC'    ;
                           'Bell-5170LR'  3       100      200000       0        20000        2           0             0.25
0        5      'DC'    ;
                           'Bell-5170MR'  3      1000     2000000       0        20000        2           0             0.25
0        5      'DC'    ;
                           'Bell-5170HR'  3     10000    20000000       0        20000        2           0             0.25
0        5      'DC'    ;
                           'TAM - 1'      3       0.0        1000       0           60        1           0              0
10.0      36      'DC'    ;
                           'TAM - 2'      3       0.0        1000       0           60        1           0              0
10.0      38      'DC'    ;
                           'HMR2300'      3       0.1        2000       0            0      0.52          0           0.0065
0       15      'DC'    };

ACFieldUnitLabelStr1 = {  '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
WEIGHT     ELECT WEIGHT  ]
%    Units             [   Low deg C High     Low deg C High          L       W       H         L       W       H         grams
grams      ]
AC_FIELD_SENSOR_DB2 = {     0          0         0          0         6.096   11.938   2.540    6.096   11.938   2.540     141.748
141.748     ;
                            0          0         0          0         5.000    3.700   2.700    8.900   17.100   3.600       0
0        ;
                          -10.0      50.0      -10.0      50.0        12.00    7.600   3.700    12.00    7.600   3.700      177.00
177.00      ;
                          -10.0      50.0      -10.0      50.0        12.00    7.600   3.700    12.00    7.600   3.700      177.00
177.00      ;
                            0.0      50.0      -25.0      70.0        14.047   1.811   1.811    17.526   9.906   3.658       0
0        ;
                            0.0      50.0      -25.0      70.0        15.494   0.889   0.889    17.526   9.906   3.658       0
0        ;
                            0.0      50.0      -25.0      70.0        15.494   0.889   0.889    17.526   9.906   3.658       0
0        ;
                            0.0      50.0      -25.0      70.0        15.494   0.889   0.889    17.526   9.906   3.658       0
0        ;
                          -90.0      80.0      -34.0      71.0         4.76     6.68    11.9     5.23    14.02   16.38       310.0
770.0     ;
                          -39.0      76.0      -39.0      76.0         4.45     7.62    14.3      0        0       0          500.0
0        ;
                          -40.0      85.0      -40.0      85.0        10.67     3.81    2.23      0        0       0           98.0
98.0      };

ACFieldUnitLabelStr2 = {  'deg C'  'deg C'  'deg C'  'deg C'  'cm'  'cm'  'cm'  'cm'  'cm'  'cm'  'gram'  'gram'  };

end


if strcmpi(UNITS,'BRITISH') == 1
%  Properties          [     SENSOR TEMP      ELECTRONICS TEMP       SENSOR DIMENSIONS (in)     ELECT DIMENSIONS (in)    SENS WEIGHT
ELECT WEIGHT  ]
%    Units             [    Low deg F High    Low deg F High          L       W       H         L       W       H          oz
oz       ]
AC_FIELD_SENSOR_DB2 = {     0          0         0          0         2.40    4.70    1.00     2.40    4.70    1.00       5.00
5.00        ;
                            0          0         0          0         2.00    1.50    1.00     3.50    6.75    1.40        0
0         ;
                           14.0      122.0      14.0      122.0        4.70    3.00    1.75     4.70    3.00    1.75       6.24
6.24        ;
                           14.0      122.0      14.0      122.0        4.70    3.00    1.75     4.70    3.00    1.75       6.24
6.24        ;
                           32.0      122.0      32.0      158.0        5.53    0.71    0.71     6.90    3.90    1.44        0
0        ;
                           32.0      122.0      32.0      158.0        6.10    0.35    0.35     6.90    3.90    1.44        0
0        ;
                           32.0      122.0      32.0      158.0        6.10    0.35    0.35     6.90    3.90    1.44        0
0        ;
                           32.0      122.0      32.0      158.0        6.10    0.35    0.35     6.90    3.90    1.44        0
0        ;
```

```
                        -130.0    176.0     -29.2     159.8        1.88    2.63    4.69      2.06    5.52    6.45      12.64
27.2       ;
                         -38.2    168.8     -38.2     168.8        1.75    3.00    5.63         0       0       0      17.60
0        ;
                         -40.0    185.0     -40.0     185.0        4.20    1.50   0.876         0       0       0       3.46
0        };

ACFieldUnitLabelStr2 = {  'deg F'  'deg F'  'deg F'  'deg F'  'in'  'in'  '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 and military programs (MIL-STD-975).  Fully Space Qualified';
AC_FIELD_SENSOR_DB3{10,1} = 'TAM-2 series parts meet NASA and 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('\n\n  ***  WARNING:  PLANETARY MAGNETIC FIELD NOT DEFINED!!  ***');
        fprintf('\n  Sensor Design Based on other properties!');
    end
end


c = 1;
%  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_FIELD_SENSOR_DB;
    end

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

    %  Select Via Temperature Operational Range
       c = 1;
       %  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;
        c = 1;
        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

    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
```

```
                for j = 1:size(AC_MATRIX3,2)
                    AC_MATRIX4{k,j} = AC_MATRIX3{c,j};
                end
                k = k + 1;
            end
        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;
            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
    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_FIELD_SENSOR_Final.mat', 'AC_MATRIX5','ACFieldUnitLabelStr');

%*************************************************************************************************************************
%
%   ***   PRINT SUMMARY Output Section for 'Print' = 'Y'

if strcmpi(Print,'Y') == 1

    fprintf('\n\n   ***   A C   F I E L D   S E N S O R   S U M M A R Y   R E S U L T S   ***');
    fprintf('\n\n***   I N P U T S   ***');
    fprintf('\n  Magnetic Field Strength:   %9.3f   %s', PLACFIELD, MagFieldUnit);
    fprintf('\n  Unit System:          %s', UNITS);

    fprintf('\n\n*************************************************************************\n');

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

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

    fprintf('\n\n  Physical Properties');
    fprintf('\n  Sensor Dimensions - Length:         %9.3f   %s', AC_MATRIX5{1,17}, ACFieldUnitLabelStr{17} );
    fprintf('\n                        Width:        %9.3f   %s', AC_MATRIX5{1,18}, ACFieldUnitLabelStr{18} );
    fprintf('\n                       Height:        %9.3f   %s', AC_MATRIX5{1,19}, ACFieldUnitLabelStr{19} );
    fprintf('\n  Sensor Mass:                        %9.3f   %s', AC_MATRIX5{1,23}, ACFieldUnitLabelStr{23} );
    fprintf('\n  Electronics Dimensions - Length:    %9.3f   %s', AC_MATRIX5{1,20}, ACFieldUnitLabelStr{20} );
    fprintf('\n                           Width:    %9.3f   %s', AC_MATRIX5{1,21}, ACFieldUnitLabelStr{21} );
```

```
    fprintf('\n                              Height:    %9.3f   %s', AC_MATRIX5{1,22}, ACFieldUnitLabelStr{22} );
    fprintf('\n  Sensor Mass:                           %9.3f   %s', AC_MATRIX5{1,24}, ACFieldUnitLabelStr{24} );

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

    fprintf('\n\n  Sensor Comments:');
    fprintf('\n  %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, '\n\n   ***   A C   F I E L D   S E N S O R   S U M M A R Y   R E S U L T S   ***');
    fprintf(Fout, '\n\n***   I N P U T S   ***');
    fprintf(Fout, '\n  Magnetic Field Strength:   %9.3f   %s', PLACFIELD, MagFieldUnit);
    fprintf(Fout, '\n  Unit System:          %s', UNITS);

    fprintf(Fout, '\n\n*************************************************************************\n');

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

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

    fprintf(Fout, '\n\n  Physical Properties');
    fprintf(Fout, '\n  Sensor Dimensions - Length:         %9.3f   %s', AC_MATRIX5{1,17}, ACFieldUnitLabelStr{17} );
    fprintf(Fout, '\n                       Width:         %9.3f   %s', AC_MATRIX5{1,18}, ACFieldUnitLabelStr{18} );
    fprintf(Fout, '\n                      Height:         %9.3f   %s', AC_MATRIX5{1,19}, ACFieldUnitLabelStr{19} );
    fprintf(Fout, '\n  Sensor Mass:                        %9.3f   %s', AC_MATRIX5{1,23}, ACFieldUnitLabelStr{23} );
    fprintf(Fout, '\n  Electronics Dimensions - Length:    %9.3f   %s', AC_MATRIX5{1,20}, ACFieldUnitLabelStr{20} );
    fprintf(Fout, '\n                           Width:    %9.3f   %s', AC_MATRIX5{1,21}, ACFieldUnitLabelStr{21} );
    fprintf(Fout, '\n                          Height:    %9.3f   %s', AC_MATRIX5{1,22}, ACFieldUnitLabelStr{22} );
```

```
    fprintf(Fout, '\n  Sensor Mass:                          %9.3f   %s', AC_MATRIX5{1,24}, ACFieldUnitLabelStr{24} );

    fprintf(Fout, '\n\n  Power Requirements');
    fprintf(Fout, '\n  Input Voltage:                %9.3f   %s', AC_FIELD_SENSOR_DB{1,11}, ACFieldUnitLabelStr{11} );
    fprintf(Fout, '\n  Voltage Type:                       %s   %s', AC_FIELD_SENSOR_DB{1,12}, ACFieldUnitLabelStr{12} );

    fprintf(Fout, '\n\n  Sensor Comments:');
    fprintf(Fout, '\n  %s', AC_MATRIX5{1,25});

    fclose(Fout);
```

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
%    http://www.omega.com/toc_asp/subsectionSC.asp?subsection=K02&book=Pressure
%    http://www.endevco.com/feature/Featured.aspx?page=71
%    http://www.bksv.com/pdf/2271A_AM20.pdf
%    http://www.honeywell.com/sites/servlet/com.merx.npoint.servlets.DocumentServlet?docid=D732BF1BD-AFCB-3204-9D96-
2292C529E86F
%

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        LINEARITY
RESONANCE FREQ.    SHOCK LIMITS  VIB LIMINTS       INPUT          CURRENT RANGE       BIAS VOLTAGE    ELECT NOISE ]
 %     Units                    [       TYPE     Low  g's  High      mV/G                 %              Low  Hz  High          %
Hz                  g's           g's             Type           Low    mA  High      V       Type          mG's     ]
    ACCELEROMETER_SENSOR_DB1 = { 'JTF-GPA'  -500.0     500.0       0.08              5.00              0.00    2400.00        1.00
5000.00          10000.0       10000.0          'N/A'              0        0       10.0    'DC'           0       ;  %1
                                  'MA11'     -80.0      80.0      100.0              5.00              2.00   10000.00        1.00
18000.00          2000.0        2000.0     'Const Current'     0.500    8.000      12.0    'DC'           0.10    ;  %2
                                  'MA12'     -80.0      80.0      100.0              5.00              2.00    8000.00        1.00
18000.00             0             0       'Const Current'     0.500    8.000      12.0    'DC'           0.10    ;  %3
                                  'MA15'     -50.0      50.0        0                5.00              2.00    1000.00        1.00
5000.00              0             0          'Voltage'        4.000   20.000      32.0    'DC'           0.30    ;  %4
                                  'MA21'     -80.0      80.0      100.0              5.00              2.00    9000.00        1.00
18000.00             0             0       'Const Current'     0.500    8.000      12.0    'DC'           0.10    ;  %5
                                  'MA23'     -10.0      10.0     1000.0              5.00              5.00   14000.00        1.00
25000.00             0             0       'Const Current'     5.000    8.000      12.0    'DC'           0.10    ;  %6
                                 'MA311'     -50.0      50.0      100.0              5.00              1.00    1000.00        1.00
18000.00          1000.0        1000.0         'Voltage'         7.000    7.000      24.0    'DC'          10.00    ;  %7
                                 'MA321'     -50.0      50.0        1.6              5.00              1.00    1000.00        1.00
18000.00          1000.0        1000.0         'Voltage'         4.000   20.000      30.0    'DC'          10.00    ;  %8
                                 'MA331'     -50.0      50.0        0                5.00              1.00    1000.00        1.00
18000.00          1000.0        1000.0         'Voltage'         7.000    7.000       5.0    'DC'          10.00    ;  %9
                                 'MA342'     -80.0      80.0      100.0              5.00              2.00   15000.00        1.00
22000.00             0             0       'Const Current'     0.500    8.000      12.0    'DC'           0.10    ;  %10
                                 'MAQ13'    -800.0     800.0       20.0              5.00               0         0          1.00
30000.00             0             0     'Self Generating'       0        0         0     'Charge'         0       ;  %11
                                 'MAQ14'   -1000.0    1000.0       50.0              5.00              1.00   10000.00        1.00
18000.00             0             0     'Self Generating'       0        0         0     'Charge'         0       ;  %12
                                 'MAQ36'   -2000.0    2000.0        5.0              5.00              1.00   30000.00        1.00
45000.00             0             0     'Self Generating'       0        0         0     'Charge'         0       ;  %13
                                 'MAQ41'    -100.0     100.0     1100.0              5.00               0         0          1.00
10000.00             0             0     'Self Generating'       0        0         0     'Charge'         0       ;  %14
```

```
    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_FIELD_SENSOR_Final.mat', 'AC_MATRIX5','ACFieldUnitLabelStr');

%*************************************************************************************************************************
%
%   ***   PRINT SUMMARY Output Section for 'Print' = 'Y'

if strcmpi(Print,'Y') == 1

    fprintf('\n\n   ***   A C   F I E L D   S E N S O R   S U M M A R Y   R E S U L T S   ***');
    fprintf('\n\n***   I N P U T S   ***');
    fprintf('\n  Magnetic Field Strength:   %9.3f   %s', PLACFIELD, MagFieldUnit);
    fprintf('\n  Unit System:          %s', UNITS);

    fprintf('\n\n*************************************************************************\n');

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

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

    fprintf('\n\n  Physical Properties');
    fprintf('\n  Sensor Dimensions - Length:         %9.3f   %s', AC_MATRIX5{1,17}, ACFieldUnitLabelStr{17} );
    fprintf('\n                        Width:        %9.3f   %s', AC_MATRIX5{1,18}, ACFieldUnitLabelStr{18} );
    fprintf('\n                       Height:        %9.3f   %s', AC_MATRIX5{1,19}, ACFieldUnitLabelStr{19} );
    fprintf('\n  Sensor Mass:                        %9.3f   %s', AC_MATRIX5{1,23}, ACFieldUnitLabelStr{23} );
    fprintf('\n  Electronics Dimensions - Length:    %9.3f   %s', AC_MATRIX5{1,20}, ACFieldUnitLabelStr{20} );
    fprintf('\n                           Width:    %9.3f   %s', AC_MATRIX5{1,21}, ACFieldUnitLabelStr{21} );
```

```
    fprintf('\n                              Height:    %9.3f   %s', AC_MATRIX5{1,22}, ACFieldUnitLabelStr{22} );
    fprintf('\n  Sensor Mass:                           %9.3f   %s', AC_MATRIX5{1,24}, ACFieldUnitLabelStr{24} );

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

    fprintf('\n\n  Sensor Comments:');
    fprintf('\n  %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, '\n\n   ***   A C   F I E L D   S E N S O R   S U M M A R Y   R E S U L T S   ***');
    fprintf(Fout, '\n\n***   I N P U T S   ***');
    fprintf(Fout, '\n  Magnetic Field Strength:   %9.3f   %s', PLACFIELD, MagFieldUnit);
    fprintf(Fout, '\n  Unit System:          %s', UNITS);

    fprintf(Fout, '\n\n*************************************************************************\n');

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

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

    fprintf(Fout, '\n\n  Physical Properties');
    fprintf(Fout, '\n  Sensor Dimensions - Length:         %9.3f   %s', AC_MATRIX5{1,17}, ACFieldUnitLabelStr{17} );
    fprintf(Fout, '\n                       Width:         %9.3f   %s', AC_MATRIX5{1,18}, ACFieldUnitLabelStr{18} );
    fprintf(Fout, '\n                      Height:         %9.3f   %s', AC_MATRIX5{1,19}, ACFieldUnitLabelStr{19} );
    fprintf(Fout, '\n  Sensor Mass:                        %9.3f   %s', AC_MATRIX5{1,23}, ACFieldUnitLabelStr{23} );
    fprintf(Fout, '\n  Electronics Dimensions - Length:    %9.3f   %s', AC_MATRIX5{1,20}, ACFieldUnitLabelStr{20} );
    fprintf(Fout, '\n                           Width:    %9.3f   %s', AC_MATRIX5{1,21}, ACFieldUnitLabelStr{21} );
    fprintf(Fout, '\n                          Height:    %9.3f   %s', AC_MATRIX5{1,22}, ACFieldUnitLabelStr{22} );
```

```
    fprintf(Fout, '\n  Sensor Mass:                          %9.3f   %s', AC_MATRIX5{1,24}, ACFieldUnitLabelStr{24} );

    fprintf(Fout, '\n\n  Power Requirements');
    fprintf(Fout, '\n  Input Voltage:                %9.3f   %s', AC_FIELD_SENSOR_DB{1,11}, ACFieldUnitLabelStr{11} );
    fprintf(Fout, '\n  Voltage Type:                       %s   %s', AC_FIELD_SENSOR_DB{1,12}, ACFieldUnitLabelStr{12} );

    fprintf(Fout, '\n\n  Sensor Comments:');
    fprintf(Fout, '\n  %s', AC_MATRIX5{1,25});

    fclose(Fout);
```

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
%    http://www.omega.com/toc_asp/subsectionSC.asp?subsection=K02&book=Pressure
%    http://www.endevco.com/feature/Featured.aspx?page=71
%    http://www.bksv.com/pdf/2271A_AM20.pdf
%    http://www.honeywell.com/sites/servlet/com.merx.npoint.servlets.DocumentServlet?docid=D732BF1BD-AFCB-3204-9D96-
2292C529E86F
%

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        LINEARITY
RESONANCE FREQ.    SHOCK LIMITS  VIB LIMINTS       INPUT          CURRENT RANGE       BIAS VOLTAGE    ELECT NOISE ]
 %     Units                    [       TYPE     Low  g's  High      mV/G                 %              Low  Hz  High          %
Hz                  g's           g's             Type           Low    mA  High      V       Type          mG's     ]
    ACCELEROMETER_SENSOR_DB1 = { 'JTF-GPA'  -500.0     500.0       0.08              5.00              0.00    2400.00        1.00
5000.00          10000.0       10000.0          'N/A'              0        0       10.0    'DC'           0       ;  %1
                                  'MA11'     -80.0      80.0      100.0              5.00              2.00   10000.00        1.00
18000.00          2000.0        2000.0     'Const Current'     0.500    8.000      12.0    'DC'           0.10    ;  %2
                                  'MA12'     -80.0      80.0      100.0              5.00              2.00    8000.00        1.00
18000.00             0             0       'Const Current'     0.500    8.000      12.0    'DC'           0.10    ;  %3
                                  'MA15'     -50.0      50.0        0                5.00              2.00    1000.00        1.00
5000.00              0             0          'Voltage'        4.000   20.000      32.0    'DC'           0.30    ;  %4
                                  'MA21'     -80.0      80.0      100.0              5.00              2.00    9000.00        1.00
18000.00             0             0       'Const Current'     0.500    8.000      12.0    'DC'           0.10    ;  %5
                                  'MA23'     -10.0      10.0     1000.0              5.00              5.00   14000.00        1.00
25000.00             0             0       'Const Current'     5.000    8.000      12.0    'DC'           0.10    ;  %6
                                 'MA311'     -50.0      50.0      100.0              5.00              1.00    1000.00        1.00
18000.00          1000.0        1000.0         'Voltage'         7.000    7.000      24.0    'DC'          10.00    ;  %7
                                 'MA321'     -50.0      50.0        1.6              5.00              1.00    1000.00        1.00
18000.00          1000.0        1000.0         'Voltage'         4.000   20.000      30.0    'DC'          10.00    ;  %8
                                 'MA331'     -50.0      50.0        0                5.00              1.00    1000.00        1.00
18000.00          1000.0        1000.0         'Voltage'         7.000    7.000       5.0    'DC'          10.00    ;  %9
                                 'MA342'     -80.0      80.0      100.0              5.00              2.00   15000.00        1.00
22000.00             0             0       'Const Current'     0.500    8.000      12.0    'DC'           0.10    ;  %10
                                 'MAQ13'    -800.0     800.0       20.0              5.00               0         0          1.00
30000.00             0             0     'Self Generating'       0        0         0     'Charge'         0       ;  %11
                                 'MAQ14'   -1000.0    1000.0       50.0              5.00              1.00   10000.00        1.00
18000.00             0             0     'Self Generating'       0        0         0     'Charge'         0       ;  %12
                                 'MAQ36'   -2000.0    2000.0        5.0              5.00              1.00   30000.00        1.00
45000.00             0             0     'Self Generating'       0        0         0     'Charge'         0       ;  %13
                                 'MAQ41'    -100.0     100.0     1100.0              5.00               0         0          1.00
10000.00             0             0     'Self Generating'       0        0         0     'Charge'         0       ;  %14
```

```
                              'MAT53'      -80.0      80.0      100.0            5.00               2.00   10000.00       1.00
18000.00          0             0        'Const Current'    0.500    8.000     12.0    'DC'          0.30     ;  %15
                                'PA'     -1000.0   1000.0       5.0            5.00               3.00    5000.00         0
30000.00        3000.0        2000.0        'N/A'             0        0       16.5    'DC'          0      ;  %16
                              'ACC301'     -500.0    500.0      10.0            5.00               3.00   10000.00       2.00
50000.00        5000.0          0.0      'Const Current'    2.000    2.000     30.0    'DC'          1.50     ;  %17
                             'OM-CP-EB'    -100.0    100.0        0               0                0.00     400.00         0
0               0               0        'N/A'             0         0       9.0   'DC'          0      ;  %18
                              '71-60K'   -60000.0  60000.0    3000.0            5.00               0.00  100000.00       2.00
700000.00       120000.0        0           'Voltage'          0        0       10.0    'DC'          0      ;  %19
                            '2271A/AM20'      0.0  10000.0        0             3.00               0.50    7000.00       1.00
27000.00        10000.0       1000.0    'Self Generating'     0        0         0    'Charge'      0.30     ;  %20
                              '7264D'     -2000.0   2000.0      0.20            1.00              20.00    6000.00       1.00
40000.00        10000.0       1000.0       'Voltage'           0        0       10.0    'DC'          0      ;  %21
                            'QA2000-030'     -60.0     60.0        0               0               20.00    2000.00         0
0               250.0          15.0      'Const Current'  16.000   16.000     28.0   'DC'          0.30      };   %22

    AccUnitLabelStr1 = { 'N/A'  'g''s'  'g''s'  'mV/g'  '%'  'Hz'  'Hz'   '%'  'Hz'  'g''s'  'g''s'  'Type'   'mA'   'mA'  'V'
'Type'  'mG''s' };

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%   Accelerometer Sensor Data Base Created Based on un Unit System
%%   2 - D Matrix  ACCELEROMETER_SENSOR_DB(A_Type, Property)

if strcmpi(UNITS,'SI') == 1
 %  Properties             [        PROBE TEMP RANGE       WEIGHT      DIMENSIONS PROBE - cm        TEMP SENS       POWER  ]
 %    Units                [        Low  deg C  High         gr        Length     Width    Height   g's/deg C         W     ]
    ACCELEROMETER_SENSOR_DB2 = {    21.1         93.3       129.3      3.556     3.556    3.556        0              0     ;  %1
                                   -56.7        137.8       150.3      6.706     2.489    2.489      0.261            0     ;  %2
                                   -56.7        137.8       170.1      5.283     2.896    2.540      0.261            0     ;  %3
                                   -23.3         98.9       141.7      5.690     2.896    2.896      0.261          0.640   ;  %4
                                   -12.2        137.8       110.6      4.140     1.702    1.702      0.261            0     ;  %5
                                   -56.7        121.1         8.5      1.397     1.397    1.397      0.261            0     ;  %6
                                   -17.8         82.2        39.7      2.540     2.540    2.540      0.261          0.168   ;  %7
                                   -17.8         82.2        39.7      2.540     2.540    2.540      0.261            0     ;  %8
                                   -17.8         82.2        39.7      2.540     2.540    2.540      0.261            0     ;  %9
                                   -23.3        137.8        28.3      4.089     1.880    1.880      0.261            0     ;  %10
                                   -56.7        248.9        28.3      2.388     1.500    1.500      0.261            0     ;  %11
                                   -56.7        248.9        34.0      2.007     1.905    1.905      0.261            0     ;  %12
                                   -56.7        248.9         7.1      1.194     0.838    0.838      0.261            0     ;  %13
                                   -56.7        248.9        99.2      4.191     3.200    3.200      0.261            0     ;  %14
                                   -23.3        137.8       107.7      5.004     2.489    2.489      0.261            0     ;  %15
                                     0.0         71.1        85.0      4.064     2.540    2.540        0              0     ;  %16
                                   -55.0        120.0        10.0      2.030     2.030    2.030        0              0     ;  %17
                                   -20.0         54.0      2300.0      8.900    11.100    2.600        0              0     ;  %18
                                   -54.0        121.1         0.06     0.635     0.635    0.635     72.000            0     ;  %19
                                  -269.0        260.0        27.0      2.920     1.588    1.588        0              0     ;  %20
                                   -18.0         66.0         1.0      1.143     1.016    0.513      2.000            0     ;  %21
                                   -55.0         95.0        71.0      3.350     3.350    2.718      0.0003         0.480   };  %22

    AccUnitLabelStr2 = {  'deg C'  'deg C'   'gr'   'cm'   'cm'   'cm'   'g''s/deg C'  'W'    };

end
```

```
if strcmpi(UNITS,'British') == 1
 %  Properties              [        PROBE TEMP RANGE        WEIGHT      DIMENSIONS PROBE - in      TEMP SENS       POWER    ]
 %    Units                 [        Low  deg F  High          oz        Length    Width    Height   g's/deg F      BTUs/hr  ]
    ACCELEROMETER_SENSOR_DB2 = {     70.0          200.0        4.560     1.400     1.400     1.400        0             0     ;  %1
                                    -70.0          280.0        5.300     2.640     0.980     0.980      0.145           0     ;  %2
                                    -70.0          280.0        6.000     2.080     1.140     1.000      0.145           0     ;  %3
                                    -10.0          210.0        5.000     2.240     1.140     1.140      0.145         2.186   ;  %4
                                     10.0          280.0        3.900     1.630     0.670     0.670      0.145           0     ;  %5
                                    -70.0          250.0        0.300     0.550     0.550     0.550      0.145           0     ;  %6
                                      0.0          180.0        1.400     1.000     1.000     1.000      0.145         0.574   ;  %7
                                      0.0          180.0        1.400     1.000     1.000     1.000      0.145           0     ;  %8
                                      0.0          180.0        1.400     1.000     1.000     1.000      0.145           0     ;  %9
                                    -10.0          280.0        1.000     1.610     0.740     0.740      0.145           0     ;  %10
                                    -70.0          480.0        1.000     0.940     0.590     0.590      0.145           0     ;  %11
                                    -70.0          480.0        1.200     0.790     0.750     0.750      0.145           0     ;  %12
                                    -70.0          480.0        0.250     0.470     0.330     0.330      0.145           0     ;  %13
                                    -70.0          480.0        3.500     1.650     1.260     1.260      0.145           0     ;  %14
                                    -10.0          280.0        3.800     1.970     0.980     0.980      0.145           0     ;  %15
                                     32.0          160.0        3.000     1.600     1.000     1.000        0             0     ;  %16
                                    -65.0          250.0        0.350     0.800     0.800     0.800        0             0     ;  %17
                                     -4.0          129.0       80.000     3.500     4.400     1.000        0             0     ;  %18
                                    -65.0          250.0        0.002     0.250     0.250     0.250     40.000           0     ;  %19
                                   -452.0          500.0        0.950     1.150     0.625     0.625        0             0     ;  %20
                                      0.0          150.0        0.035     0.400     0.400     0.202      1.200           0     ;  %21
                                    -67.0          203.0        2.504     1.320     1.320     1.070     0.00017        1.640   };
%22

    AccUnitLabelStr2 = {  'deg F'  'deg F'   'oz'   'in'  'in'  'in'   'g''s/deg F'   'BTUs/hr'   };

end

%  Combine Data Arrays to a Single Matrix
ACCELEROMETER_SENSOR_DB = cat(2,ACCELEROMETER_SENSOR_DB1,ACCELEROMETER_SENSOR_DB2);
AccUnitLabelStr = cat(2,AccUnitLabelStr1,AccUnitLabelStr2);


%  Additional Comments on Sensor
ACCELEROMETER_SENSOR_DB{ 1,26} = '3 JTF Units mounted on Triaxial Mounting block Records 3 axis data simultaneously.';
ACCELEROMETER_SENSOR_DB{ 2,26} = 'General Purpose.';
ACCELEROMETER_SENSOR_DB{ 3,26} = 'General Purpose.';
ACCELEROMETER_SENSOR_DB{ 4,26} = 'General Purpose. 4 - 20 mA Proportional to G.';
ACCELEROMETER_SENSOR_DB{ 5,26} = 'General Purpose. Miniature Size, Low Profile.  Sensitivity up to 1000mV/G special order.';
ACCELEROMETER_SENSOR_DB{ 6,26} = 'Miniature Size, Low Profile.  Sensitivity down to 100mV/G special order.';
ACCELEROMETER_SENSOR_DB{ 7,26} = 'DC response.  Amplified Voltage Output.  Range +/-1, 2, 5, 10, 20, 50 g''s available';
ACCELEROMETER_SENSOR_DB{ 8,26} = '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_DB{ 9,26} = '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_DB{10,26} = 'High Frequency. Sensitivity up to 1000mV/G special order.';
ACCELEROMETER_SENSOR_DB{11,26} = '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_DB{12,26} = '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 sets Frequency Range.
Sensitivity in pC/G.';
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 5VDC 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.';
ACCELEROMETER_SENSOR_DB{21,26} = 'Single Axis, piezoresistive accelerometer.  Small size, rugged design. Ideal for long
duration transient shocks.  Flight Proven - Huygens Lander Penetrometer SSP.';
ACCELEROMETER_SENSOR_DB{22,26} = 'Single Axis, piezoresistive accelerometer.  Long-term repeatability and superior reliability.
Dual built-in self-test.  Flight Proven - Huygens Lander Descent phase HASI.';



%  Select Acceleration Sensor by Input Operation Range Value
c = 1;
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};
            end
            c = c + 1;
        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};
            end
            c = c + 1;
        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};
            end
            c = c + 1;
        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};
```

```
            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
            c = c + 1;
        end
    end
end

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

    %  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 Accelerometer 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
            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

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


if size(ACC_MATRIX4,1) >1

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

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


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

```
            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
        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
    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

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('\n\n   ***   A C C E L E R O M E T E R   S E N S O R   S U M M A R Y   R E S U L T S   ***');
    fprintf('\n\n***   I N P U T S   ***');
    fprintf('\n  Mission Acceleration Profile:     %s', char(ACC_TYPE));
    fprintf('\n  Accelerometer Power Type:         %s', char(POW_TYPE));
    fprintf('\n  Unit System:           %s', UNITS);

    fprintf('\n\n*************************************************************************\n');

    %  Accelerometer Sensor Properties
```

```
    fprintf('\n\n** Sensor Properties **');
    fprintf('\n  Accelerometer Sensor Type:                    %s', ACC_MATRIX6{1,1});
    fprintf('\n  Accelerometer Sensing Range - Low:    %10.2f   %s', ACC_MATRIX6{1,2}, AccUnitLabelStr{2} );
    fprintf('\n                                High:    %10.2f   %s', ACC_MATRIX6{1,3}, AccUnitLabelStr{3} );
    fprintf('\n  Sensor Sensitivity:                   %10.2f   %s', ACC_MATRIX6{1,4}, AccUnitLabelStr{4} );
    fprintf('\n  Transverse Sensitivity:               %10.2f   %s', ACC_MATRIX6{1,5}, AccUnitLabelStr{5} );
    fprintf('\n  Operating Frequency Range - Low:      %10.2f   %s', ACC_MATRIX6{1,6}, AccUnitLabelStr{6} );
    fprintf('\n                              High:     %10.2f   %s', ACC_MATRIX6{1,7}, AccUnitLabelStr{7} );
    fprintf('\n  Sensor Linearity:                     %10.2f   %s', ACC_MATRIX6{1,8}, AccUnitLabelStr{8} );
    fprintf('\n  Resonance Frequency:                  %10.2f   %s', ACC_MATRIX6{1,9}, AccUnitLabelStr{9} );
    fprintf('\n  Shock Limits:                         %10.2f   %s', ACC_MATRIX6{1,10}, AccUnitLabelStr{10} );
    fprintf('\n  Vibration Limits:                     %10.2f   %s', ACC_MATRIX6{1,11}, AccUnitLabelStr{11} );


    fprintf('\n\n  Power Requirements');
    fprintf('\n  Input Type:                                    %s   %s', ACC_MATRIX6{1,12}, AccUnitLabelStr{12} );
    fprintf('\n  Current Range - Low:                  %10.2f   %s', ACC_MATRIX6{1,13}, AccUnitLabelStr{13} );
    fprintf('\n                 High:                  %10.2f   %s', ACC_MATRIX6{1,14}, AccUnitLabelStr{14} );
    fprintf('\n  Bias Voltage:                         %10.2f   %s', ACC_MATRIX6{1,15}, AccUnitLabelStr{15} );
    fprintf('\n  Bias Voltage Type:                             %s   %s', ACC_MATRIX6{1,16}, AccUnitLabelStr{16} );
    fprintf('\n  Electrical Noise:                     %10.2f   %s', ACC_MATRIX6{1,17}, AccUnitLabelStr{17} );
    fprintf('\n  Electrical Power:                     %10.2f   %s', ACC_MATRIX6{1,25}, AccUnitLabelStr{25} );

    %  Environmental Properties
    fprintf('\n\n   Environmental Properties');
    fprintf('\n  Sensor Temperature Range - Low:       %10.2f   %s', ACC_MATRIX6{1,18}, AccUnitLabelStr{18} );
    fprintf('\n                             High:      %10.2f   %s', ACC_MATRIX6{1,19}, AccUnitLabelStr{19} );
    fprintf('\n  Temperature Sensitivity:              %10.3f   %s', ACC_MATRIX6{1,24}, AccUnitLabelStr{24} );


    fprintf('\n\n  Physical Properties');
    fprintf('\n  Sensor Mass:                         %9.3f   %s', ACC_MATRIX6{1,20}, AccUnitLabelStr{20} );
    fprintf('\n  Sensor Dimensions - Length:          %9.3f   %s', ACC_MATRIX6{1,21}, AccUnitLabelStr{21} );
    fprintf('\n                       Width:          %9.3f   %s', ACC_MATRIX6{1,22}, AccUnitLabelStr{22} );
    fprintf('\n                      Height:          %9.3f   %s', ACC_MATRIX6{1,23}, AccUnitLabelStr{23} );


    fprintf('\n\n  Sensor Comments:');
    fprintf('\n  %s', ACC_MATRIX6{1,26});

end


%*************************************************************************
%***     Writes a text file summary of the Planetary Properties

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

    fprintf(Fout, '\n\n   ***   A C C E L E R O M E T E R   S E N S O R   S U M M A R Y   R E S U L T S   ***');
    fprintf(Fout, '\n\n***   I N P U T S   ***');
    fprintf(Fout, '\n  Mission Acceleration Profile:     %s', char(ACC_TYPE));
    fprintf(Fout, '\n  Accelerometer Power Type:         %s', char(POW_TYPE));
    fprintf(Fout, '\n  Unit System:          %s', UNITS);
```

```
fprintf(Fout, '\n\n*********************************************************************\n');

%  Accelerometer Sensor Properties
fprintf(Fout, '\n\n** Sensor Properties **');
fprintf(Fout, '\n  Accelerometer Sensor Type:                %s', ACC_MATRIX6{1,1});
fprintf(Fout, '\n  Accelerometer Sensing Range - Low:    %10.2f   %s', ACC_MATRIX6{1,2}, AccUnitLabelStr{2} );
fprintf(Fout, '\n                              High:    %10.2f   %s', ACC_MATRIX6{1,3}, AccUnitLabelStr{3} );
fprintf(Fout, '\n  Sensor Sensitivity:                   %10.2f   %s', ACC_MATRIX6{1,4}, AccUnitLabelStr{4} );
fprintf(Fout, '\n  Transverse Sensitivity:               %10.2f   %s', ACC_MATRIX6{1,5}, AccUnitLabelStr{5} );
fprintf(Fout, '\n  Operating Frequency Range - Low:      %10.2f   %s', ACC_MATRIX6{1,6}, AccUnitLabelStr{6} );
fprintf(Fout, '\n                             High:      %10.2f   %s', ACC_MATRIX6{1,7}, AccUnitLabelStr{7} );
fprintf(Fout, '\n  Sensor Linearity:                     %10.2f   %s', ACC_MATRIX6{1,8}, AccUnitLabelStr{8} );
fprintf(Fout, '\n  Resonance Frequency:                  %10.2f   %s', ACC_MATRIX6{1,9}, AccUnitLabelStr{9} );
fprintf(Fout, '\n  Shock Limits:                         %10.2f   %s', ACC_MATRIX6{1,10}, AccUnitLabelStr{10} );
fprintf(Fout, '\n  Vibration Limits:                     %10.2f   %s', ACC_MATRIX6{1,11}, AccUnitLabelStr{11} );


fprintf(Fout, '\n\n  Power Requirements');
fprintf(Fout, '\n  Input Type:                                  %s   %s', ACC_MATRIX6{1,12}, AccUnitLabelStr{12} );
fprintf(Fout, '\n  Current Range - Low:                  %10.2f   %s', ACC_MATRIX6{1,13}, AccUnitLabelStr{13} );
fprintf(Fout, '\n                  High:                %10.2f   %s', ACC_MATRIX6{1,14}, AccUnitLabelStr{14} );
fprintf(Fout, '\n  Bias Voltage:                         %10.2f   %s', ACC_MATRIX6{1,15}, AccUnitLabelStr{15} );
fprintf(Fout, '\n  Bias Voltage Type:                           %s   %s', ACC_MATRIX6{1,16}, AccUnitLabelStr{16} );
fprintf(Fout, '\n  Electrical Noise:                     %10.2f   %s', ACC_MATRIX6{1,17}, AccUnitLabelStr{17} );
fprintf(Fout, '\n  Electrical Power:                     %10.2f   %s', ACC_MATRIX6{1,25}, AccUnitLabelStr{25} );


%  Environmental Properties
fprintf(Fout, '\n\n   Environmental Properties');
fprintf(Fout, '\n  Sensor Temperature Range - Low:       %10.2f   %s', ACC_MATRIX6{1,18}, AccUnitLabelStr{18} );
fprintf(Fout, '\n                             High:      %10.2f   %s', ACC_MATRIX6{1,19}, AccUnitLabelStr{19} );
fprintf(Fout, '\n  Temperature Sensitivity:              %10.3f   %s', ACC_MATRIX6{1,24}, AccUnitLabelStr{24} );


fprintf(Fout, '\n\n  Physical Properties');
fprintf(Fout, '\n  Sensor Mass:                          %9.3f   %s', ACC_MATRIX6{1,20}, AccUnitLabelStr{20} );
fprintf(Fout, '\n  Sensor Dimensions - Length:           %9.3f   %s', ACC_MATRIX6{1,21}, AccUnitLabelStr{21} );
fprintf(Fout, '\n                       Width:           %9.3f   %s', ACC_MATRIX6{1,22}, AccUnitLabelStr{22} );
fprintf(Fout, '\n                      Height:           %9.3f   %s', ACC_MATRIX6{1,23}, AccUnitLabelStr{23} );


fprintf(Fout, '\n\n  Sensor Comments:');
fprintf(Fout, '\n  %s', ACC_MATRIX6{1,26});

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.bksv.com/pdf/bp2016.pdf
%     http://www.bksv.com/pdf/bp1803.pdf
%     http://www.bksv.com/pdf/bp1844.pdf
%     http://www.pcb.com/Linked_Documents/Vibration/Mics_1006.pdf
%     http://www.microflown.com/data/manuals_datasheets/probes/Datasheet_PU-mini_v1.0.pdf
%     http://www.pacndt.com/index.aspx?go=products&focus=/sensors/general_purpose.htm
%

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 un Unit System
%%  2 - D Matrix  ACOUSTIC_DB(A_Type, Property)

%  Acoustic Sensor Common Properties
% Properties          [    SENSOR          DYNAMIC RANGE        FREQUENCY RANGE      SENSITIVITY @250 HZ       RESONANCE
INHERENT NOISE   CLIPPING LIMIT    VOLTAGE SUPPLY   NOM CURRENT        STABILITY       OPERATIONAL    ]
% Units              [       #            Low   dB   High      Low    Hz   High     Range dB Low  Tol High      Hz
dB                dB          Min    V   Max       mA        dB/yr  Temp C  % RH       Type       ]
ACOUSTICSENSOR_DB1 = {  'B&K Type 4948'    55         160         5         20000      -57     -3        2       24000
58                165           22          30          4         0.001    20     50    'SENSOR'      ;
                        'B&K Type 4935'    30         140       100           5000      -45     -3        3          0
35                140           0            0          0         0        0      0    'SENSOR'      ;
                        'B&K Type 4938'    30         172         4          70000      -56     -3        3       60000
37                172          200         200          0         0.001    20     0    'SENSOR'      ;
```

```
                 'PCB 377B01'      28       165      4       80000      0     -2       2        0
28          165             0       0        20       0       0      0    'SENSOR'    ;
                 'PCB 377B02'      15       146      3.15    20000      0     -2       2        0
15          146             0       0        20       0       0      0    'SENSOR'    ;
                 'PCB 377B10'      30       170      4       70000      0     -2       2        0
30          170             0       0        20       0       0      0    'SENSOR'    ;
                 'PCB 377B11'      15       146      3.15    10000      0     -2       2        0
15          146             0       0        20       0       0      0    'SENSOR'    ;
                 'PCB 377A12'      30       187      4       20000      0     -2       2        0
30          187             0       0        20       0       0      0    'SENSOR'    ;
                 'PCB 377B20'      15       146      3.15    12500      0     -2       2        0
15          146             0       0        20       0       0      0    'SENSOR'    ;
                 'PCB 377B40'      20       160      4       40000      0     -2       2        0
20          160           200     200        0        0       0      0    'SENSOR'    ;
                 'PCB 377B41'      15       146      3.15    20000      0     -2       2        0
15          146           200     200        0        0       0      0    'SENSOR'    ;
                 'PCB 377A42'      10       146      2.60    20000      0     -2       2        0
10          146           200     200        0        0       0      0    'SENSOR'    ;
                 'PCB 377A50'      40       178      6.50    140000     0     -2       2        0
40          178           200     200        0        0       0      0    'SENSOR'    ;
                 'PCB 377A51'      31       192      10.0    20000      0     -2       2        0
31          192           200     200        0        0       0      0    'SENSOR'    ;
                 'PCB 377A53'      10       146      2.60     8000      0     -2       2        0
10          146           200     200        0        0       0      0    'SENSOR'    ;
                 'PCB 377A54'      15       146      3.15    10000      0     -2       2        0
15          146           200     200        0        0       0      0    'SENSOR'    ;
                 'PCB 377A60'      15       146      3.15    10000      0     -2       2        0
15          146           200     200        0        0       0      0    'SENSOR'    ;
                  'CT-190M'        56.8     135      10       2000      0      0       0      500000
56.8        135            10      15        0        0       0      0    'SENSOR'    ;
                  'PU-Mini'        0        125      1.00    20000      0      0       0        0
10          135            12      12        0        0       0      0    'VELOCITY'  ;
               'Array Type 4935'   35       140      100      5000      45    -2       2        0
35          140            28      28        3        0.3     40     90   'ARRAY'     ;
                 'USP-Mini'        0        135      1.00    20000      0      0       0        0
0          135              0       0        0        0       0      0   'ARRAY'     ;
                 'R80-Alpha'       30        62      200000  1000000    0      0       0      800000
58          120             0       0        0        0       0      0    'VELOCITY'  };

AcousticUnitLabelStr1  = { 'N/A'  'dB'  'dB' 'Hz' 'Hz' 'dB' 'dB' 'dB' 'Hz' 'dB' 'dB'  'V'  'V'  'mA'  'dB/yr'  'deg C'   '%RH'
'N/A'   };

if strcmpi(UNITS,'SI') == 1
%  Properties          [   OPERATING TEMP     STORAGE TEMP    TEMP COEFFICIENT    PRESSURE COEFFICIENT     DIMENSIONS (mm)
WEIGHT      ]
%    Units             [   Low deg C High    Low deg C High   @ 250 Hz dB/deg C   @ 250 Hz dB/kPa          L      W       H
grams       ]
    ACOUSTICSENSOR_DB2 = {  -50.0     100.0    -30.0     70.0       0.0130              -0.007            20     20     2.5
2.3        ;
                            -10.0      55.0    -25.0     70.0       0                    0                 7      7    65.0
10.0        ;
                            -40.0     300.0    -30.0     70.0       0.0030              -0.003             7      7    10.5
0         ;
```

```
                                -40.0     150.0    -40.0    150.0          0                  0                6.35   6.35     0
0         ;
                                -40.0     120.0    -40.0    120.0          0                  0                12.7   12.7     0
0         ;
                                -40.0     150.0    -40.0    150.0          0                  0                6.35   6.35     0
0         ;
                                -40.0     120.0    -40.0    120.0          0                  0                12.7   12.7     0
0         ;
                                -40.0     120.0    -40.0    120.0          0                  0                6.35   6.35     0
0         ;
                                -40.0     120.0    -40.0    120.0          0                  0                12.7   12.7     0
0         ;
                                -40.0     150.0    -40.0    150.0          0                  0                12.7   12.7     0
0         ;
                                -40.0     150.0    -40.0    150.0          0                  0                12.7   12.7     0
0         ;
                                -40.0     150.0    -40.0    150.0          0                  0                25.4   25.4     0
0         ;
                                -40.0     150.0    -40.0    150.0          0                  0                3.175  3.175    0
0         ;
                                -40.0     150.0    -40.0    150.0          0                  0                6.35   6.35     0
0         ;
                                -40.0     150.0    -40.0    150.0          0                  0                25.4   25.4     0
0         ;
                                -40.0     150.0    -40.0    150.0          0                  0                12.7   12.7     0
0         ;
                                -40.0     150.0    -40.0    150.0          0                  0                12.7   12.7     0
0         ;
                               -195.5      37.0   -195.5    120.0          0                  0                34.3   9.5    9.5
4.0        ;
                                  0.0      60.0      0.0     60.0          0                  0                40.0   12.7   12.7
25.0         ;
                                -10.0      55.0    -25.0     70.0          0                  0                65.0   7.0    7.0
10.0         ;
                                  0.0      60.0      0.0     60.0          0                  0                62.0   12.7   12.7
30.0         ;
                                -65.0     175.0    -65.0    175.0          0                  0                19.0   19.0   21.4
32.0         };

    AcousticUnitLabelStr2  = { 'deg C'  'deg C'  'deg C'  'deg C'  'dB/deg C'  'dB/kPa'  'mm'  'mm'  'mm'  'gram'   };
end

if strcmpi(UNITS,'BRITISH') == 1
%  Properties            [    OPERATING TEMP     STORAGE TEMP     TEMP COEFFICIENT    PRESSURE COEFFICIENT      DIMENSIONS (in)
WEIGHT       ]
%    Units               [   Low deg F High    Low deg F High    @ 250 Hz dB/deg F    @ 250 Hz dB/psi           L      W      H
oz          ]
    ACOUSTICSENSOR_DB2 = {  -67.0     212.0    -22.0    158.0          0.0072              -0.4826             0.79   0.79   0.10
0.08       ;
                             14.0     131.0    -13.0    158.0            0                   0                 0.25   0.25   2.60
0.35       ;
                            -40.0     572.0    -22.0    158.0          0.0016              -0.2068             0.27   0.27   0.41
0         ;
```

```
                        -40.0     302.0    -40.0     302.0          0                0              0.25    0.25    0
0       ;
                        -40.0     248.0    -40.0     248.0          0                0              0.50    0.50    0
0       ;
                        -40.0     302.0    -40.0     302.0          0                0              0.25    0.25    0
0       ;
                        -40.0     248.0    -40.0     248.0          0                0              0.50    0.50    0
0       ;
                        -40.0     248.0    -40.0     248.0          0                0              0.25    0.25    0
0       ;
                        -40.0     248.0    -40.0     248.0          0                0              0.50    0.50    0
0       ;
                        -40.0     302.0    -40.0     302.0          0                0              0.50    0.50    0
0       ;
                        -40.0     302.0    -40.0     302.0          0                0              0.50    0.50    0
0       ;
                        -40.0     302.0    -40.0     302.0          0                0              1.00    1.00    0
0       ;
                        -40.0     302.0    -40.0     302.0          0                0              0.125   0.125   0
0       ;
                        -40.0     302.0    -40.0     302.0          0                0              0.25    0.25    0
0       ;
                        -40.0     302.0    -40.0     302.0          0                0              1.00    1.00    0
0       ;
                        -40.0     302.0    -40.0     302.0          0                0              0.50    0.50    0
0       ;
                        -40.0     302.0    -40.0     302.0          0                0              0.50    0.50    0
0       ;
                       -320.0     100.0   -320.0     250.0          0                0              1.35    0.38    0.38
0.14     ;
                         32.0     140.0     32.0     140.0          0                0              1.58    0.50    0.50
0.88     ;
                         14.0     131.0    -13.0     158.0          0                0              2.6     0.25    0.25
0.35     ;
                         32.0     140.0     32.0     140.0          0                0              2.44    0.50    0.50
1.06     ;
                        -85.0     347.0    -85.0     347.0          0                0              0.75    0.75    0.84
1.13     };

    AcousticUnitLabelStr2  = { 'deg F'  'deg F'  'deg F'  'deg F'  'dB/deg F'  'dB/psi'  'in'  'in'  'in'  'oz'  };
end


%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);

tmp = 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
c = 1;
for s = 1:size(ACS_MATRIX,1)
    if (PLTEMP >= ACS_MATRIX{s,19}) && (PLTEMP <= ACS_MATRIX{s,20})
        for a = 1:size(ACS_MATRIX,2)
            ACS_MATRIX2{c,a} = ACS_MATRIX{s,a};
        end
        c = c+1;
    end
```

```
end

if exist('ACS_MATRIX2') == 0
    fprintf('\n\n  ***  WARNING  ***');
    fprintf('\n  NO VIABLE ACOUSTIC SENSORS EXISTS IN DATABASE BASED ON OPERATIONAL TEMPERATURE RANGE!');
    fprintf('\n  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
```

```
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\n   ***   A C O U S T I C   S E N S O R   S U M M A R Y   R E S U L T S   ***');
    fprintf('\n\n***   I N P U T S   ***');
    fprintf('\n  Unit System:          %s', UNITS);

    fprintf('\n\n*************************************************************************\n');

    %  Acoustic Sensor Properties
    fprintf('\n\n** Sensor Properties **');
    fprintf('\n  Acoustic Sensor Type:              %s', ACS_MATRIX4{1,1});
    fprintf('\n  Dynamic Sensing Range - Low:       %10.2f   %s', ACS_MATRIX4{1,2}, AcousticUnitLabelStr{2} );
    fprintf('\n                         High:       %10.2f   %s', ACS_MATRIX4{1,3}, AcousticUnitLabelStr{3} );
    fprintf('\n  Operating Frequency Range - Low:   %10.2f   %s', ACS_MATRIX4{1,4}, AcousticUnitLabelStr{4} );
    fprintf('\n                               High:   %10.2f   %s', ACS_MATRIX4{1,5}, AcousticUnitLabelStr{5} );
    fprintf('\n  Sensitivity at 250 Hz - Range:     %10.2f   %s', ACS_MATRIX4{1,6}, AcousticUnitLabelStr{6} );
    fprintf('\n                Tolerance - Low:     %10.2f   %s', ACS_MATRIX4{1,7}, AcousticUnitLabelStr{7} );
    fprintf('\n                           High:     %10.2f   %s', ACS_MATRIX4{1,8}, AcousticUnitLabelStr{8} );
    fprintf('\n  Resonance Frequency:               %10.2f   %s', ACS_MATRIX4{1,9}, AcousticUnitLabelStr{9} );
    fprintf('\n  Inherent Noise:                    %10.2f   %s', ACS_MATRIX4{1,10}, AcousticUnitLabelStr{10} );
    fprintf('\n  Clipping Limit:                    %10.2f   %s', ACS_MATRIX4{1,11}, AcousticUnitLabelStr{11} );
    fprintf('\n  Stability Conditions - Time:       %10.2f   %s', ACS_MATRIX4{1,15}, AcousticUnitLabelStr{15} );
    fprintf('\n                      Temperature:   %10.2f   %s', ACS_MATRIX4{1,16}, AcousticUnitLabelStr{16} );
    fprintf('\n              Relative Humidity:     %10.2f   %s', ACS_MATRIX4{1,17}, AcousticUnitLabelStr{17} );


    fprintf('\n\n  Power Requirements');
    fprintf('\n  Voltage Range - Min:                 %10.2f   %s', ACS_MATRIX4{1,12}, AcousticUnitLabelStr{12} );
    fprintf('\n                  Max:                 %10.2f   %s', ACS_MATRIX4{1,13}, AcousticUnitLabelStr{13} );
    fprintf('\n  Nominal Current:                     %10.2f   %s', ACS_MATRIX4{1,14}, AcousticUnitLabelStr{14} );


    %  Environmental Properties
    fprintf('\n\n   Environmental Properties');
    fprintf('\n  Operational Temperature Range - Low: %10.2f   %s', ACS_MATRIX4{1,19}, AcousticUnitLabelStr{19} );
    fprintf('\n                                 High: %10.2f   %s', ACS_MATRIX4{1,20}, AcousticUnitLabelStr{20} );
    fprintf('\n  Storage Temperature Range - Low:     %10.2f   %s', ACS_MATRIX4{1,21}, AcousticUnitLabelStr{21} );
```

```
    fprintf('\n                                High:     %10.2f   %s', ACS_MATRIX4{1,22}, AcousticUnitLabelStr{22} );
    fprintf('\n  Temperature Coefficient at 250 Hz:   %10.2f   %s', ACS_MATRIX4{1,23}, AcousticUnitLabelStr{23} );
    fprintf('\n  Pressure Coefficient at 250 Hz:      %10.2f   %s', ACS_MATRIX4{1,24}, AcousticUnitLabelStr{24} );


    fprintf('\n\n  Physical Properties');
    fprintf('\n  Sensor Dimensions - Length:         %9.3f   %s', ACS_MATRIX4{1,25}, AcousticUnitLabelStr{25} );
    fprintf('\n                       Width:         %9.3f   %s', ACS_MATRIX4{1,26}, AcousticUnitLabelStr{26} );
    fprintf('\n                      Height:         %9.3f   %s', ACS_MATRIX4{1,27}, AcousticUnitLabelStr{27} );
    fprintf('\n  Sensor Mass:                        %9.3f   %s', ACS_MATRIX4{1,28}, AcousticUnitLabelStr{28} );

    fprintf('\n\n  Sensor Comments:');
    fprintf('\n  %s', ACS_MATRIX4{1,29});


end


%**************************************************************************
%***     Writes a text file summary of the Planetary Properties

    Fout = fopen('ACOUSTIC_SENSOR_Summary.txt','w+');

    fprintf(Fout, '\n\n   ***   A C O U S T I C   S E N S O R   S U M M A R Y   R E S U L T S   ***');
    fprintf(Fout, '\n\n***   I N P U T S   ***');
    fprintf(Fout, '\n  Unit System:          %s', UNITS);

    fprintf(Fout, '\n\n*********************************************************************************\n');

    %  Acoustic Sensor Properties
    fprintf(Fout, '\n\n** Sensor Properties **');
    fprintf(Fout, '\n  Acoustic Sensor Type:              %s', ACS_MATRIX4{1,1});
    fprintf(Fout, '\n  Dynamic Sensing Range - Low:       %10.2f   %s', ACS_MATRIX4{1,2}, AcousticUnitLabelStr{2} );
    fprintf(Fout, '\n                          High:      %10.2f   %s', ACS_MATRIX4{1,3}, AcousticUnitLabelStr{3} );
    fprintf(Fout, '\n  Operating Frequency Range - Low:   %10.2f   %s', ACS_MATRIX4{1,4}, AcousticUnitLabelStr{4} );
    fprintf(Fout, '\n                             High:   %10.2f   %s', ACS_MATRIX4{1,5}, AcousticUnitLabelStr{5} );
    fprintf(Fout, '\n  Sensitivity at 250 Hz - Range:     %10.2f   %s', ACS_MATRIX4{1,6}, AcousticUnitLabelStr{6} );
    fprintf(Fout, '\n                Tolerance - Low:     %10.2f   %s', ACS_MATRIX4{1,7}, AcousticUnitLabelStr{7} );
    fprintf(Fout, '\n                           High:     %10.2f   %s', ACS_MATRIX4{1,8}, AcousticUnitLabelStr{8} );
    fprintf(Fout, '\n  Resonance Frequency:               %10.2f   %s', ACS_MATRIX4{1,9}, AcousticUnitLabelStr{9} );
    fprintf(Fout, '\n  Inherent Noise:                    %10.2f   %s', ACS_MATRIX4{1,10}, AcousticUnitLabelStr{10} );
    fprintf(Fout, '\n  Clipping Limit:                    %10.2f   %s', ACS_MATRIX4{1,11}, AcousticUnitLabelStr{11} );
    fprintf(Fout, '\n  Stability Conditions - Time:       %10.2f   %s', ACS_MATRIX4{1,15}, AcousticUnitLabelStr{15} );
    fprintf(Fout, '\n                       Temperature:  %10.2f   %s', ACS_MATRIX4{1,16}, AcousticUnitLabelStr{16} );
    fprintf(Fout, '\n                 Relative Humidity:  %10.2f   %s', ACS_MATRIX4{1,17}, AcousticUnitLabelStr{17} );


    fprintf(Fout, '\n\n  Power Requirements');
    fprintf(Fout, '\n  Voltage Range - Min:                %10.2f   %s', ACS_MATRIX4{1,12}, AcousticUnitLabelStr{12} );
    fprintf(Fout, '\n                  Max:                %10.2f   %s', ACS_MATRIX4{1,13}, AcousticUnitLabelStr{13} );
    fprintf(Fout, '\n  Nominal Current:                    %10.2f   %s', ACS_MATRIX4{1,14}, AcousticUnitLabelStr{14} );

    %  Environmental Properties
    fprintf(Fout, '\n\n   Environmental Properties');
```

```
    fprintf(Fout, '\n  Operational Temperature Range - Low: %10.2f   %s', ACS_MATRIX4{1,19}, AcousticUnitLabelStr{19} );
    fprintf(Fout, '\n                                 High: %10.2f   %s', ACS_MATRIX4{1,20}, AcousticUnitLabelStr{20} );
    fprintf(Fout, '\n  Storage Temperature Range - Low:     %10.2f   %s', ACS_MATRIX4{1,21}, AcousticUnitLabelStr{21} );
    fprintf(Fout, '\n                              High:     %10.2f   %s', ACS_MATRIX4{1,22}, AcousticUnitLabelStr{22} );
    fprintf(Fout, '\n  Temperature Coefficient at 250 Hz:   %10.2f   %s', ACS_MATRIX4{1,23}, AcousticUnitLabelStr{23} );
    fprintf(Fout, '\n  Pressure Coefficient at 250 Hz:      %10.2f   %s', ACS_MATRIX4{1,24}, AcousticUnitLabelStr{24} );


    fprintf(Fout, '\n\n  Physical Properties');
    fprintf(Fout, '\n  Sensor Dimensions - Length:         %9.3f   %s', ACS_MATRIX4{1,25}, AcousticUnitLabelStr{25} );
    fprintf(Fout, '\n                       Width:         %9.3f   %s', ACS_MATRIX4{1,26}, AcousticUnitLabelStr{26} );
    fprintf(Fout, '\n                      Height:         %9.3f   %s', ACS_MATRIX4{1,27}, AcousticUnitLabelStr{27} );
    fprintf(Fout, '\n  Sensor Mass:                        %9.3f   %s', ACS_MATRIX4{1,28}, AcousticUnitLabelStr{28} );

    fprintf(Fout, '\n\n  Sensor Comments:');
    fprintf(Fout, '\n  %s', ACS_MATRIX4{1,29});

    fclose(Fout);
```


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
%  http://www.mobrey.com/downloads/view_all.php?type=datasheets&classid=density
%  http://www.foxboro-eckardt.com/products/167lp_en.html
%

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       REPEATIBILITY     TEMPERATURE RANGE      TEST PRESSURE
MAX OP. PRESSURE     WEIGHT      DIMENSIONS - mm       POWER     CURRENT     VIBRATION        TOLERANCE      VISCOSITY
FLOWRATES    OPERATIONAL   ]
 %    Units          [        TYPE      Low  g/cc High     g/cc            g/cc           Low   deg C  High           kPa
kPa             kg      Len      Wid     Hgt      V DC       mA           g's            g/cc           cP            ltr/hr
Type       ]
    DENSITYSENSOR_DB =  { 'MMS 7826'    0         3       0.001          0.0001        -50.0          200.0           0
20700            0        0        0       0         25         42          0              0            500             0
'LIQUID'     ;
                      'MMS 7826L'   0         3       0.001          0.0001        -50.0          200.0           0
10000            0        0        0       0         28         45          0              0            500             0
'LIQUID'     ;
                      'MMS 7828'    0         3       0.001          0.0001        -50.0          200.0           0
20700            0        0        0       0         28         45          0              0          20000             0
'LIQUID'     ;
                      'MMS 7828L'   0         3       0.001          0.0001        -50.0          200.0           0
10000            0        0        0       0         28         45          0              0          20000             0
'LIQUID'     ;
                      'MMS 7835'    0         3      0.00015         0.00002       -50.0          110.0         22500
15000           22     1027      171     114         28         80          0.6         0.000029          0          3000
'LIQUID'     ;
                      'MMA 7835'    0         3      0.00015         0.00002       -50.0          110.0         22500
15000           22     1027      171     114         28         80          0.6         0.000029          0          3000
'LIQUID'     ;
                      'EGL 7845'    0         3       0.005          0             -50.0          160.0         15000
10000           22     1027      171     114         28         80          0            0.0006          0          3000
'LIQUID'     ;
                      'EGL 7847'    0         3       0.005          0             -50.0          160.0          3000
2000            22     1027      171     114         28         80          0            0.0006          0          3000
'LIQUID'     ;
                      'MMT 7845'    0         3      0.00035         0.00005       -50.0          110.0         15000
10000           28     1027      171     114         28         17          0            0.0006          0          3000
'GAS'      ;
                      'MMT 7846'    0         3      0.00050         0.00005       -50.0          110.0          7500
5000            28     1027      171     114         28         17          0            0.0006          0          3000
'GAS'      ;
                      'MMT 7847'    0         3      0.00035         0.00005       -50.0          110.0          3000
2000            28     1027      171     114         28         17          0            0.0006          0          3000
'GAS'      ;
```

```
                        'GDT 7812'    0.001     4       0.001           0.002        -20.0          85.0            0
15000           5       364     139     139       33          25          0             0.0006          0              216
'GAS'       ;
                        '167 LP'      0.1       1.6       0.015         0            -196.0         400.0           0
16200         18.5      362     270     210       0           0           2.0               0             0            1200
'LIQUID'     };

    DensityUnitLabelStr = { 'N/A'  'g/cc'  'g/cc'  'g/cc'  'g/cc'   'deg C'  'deg C'   'kPa'  'kPa'    'kg'   'mm'  'mm'  'mm'
'V-DC'  'mA'    'g''s'   'g/cc'    'cP'   'ltr/hr'   'Type'   };

end


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

    DENSITY_CNVT = lbft3_to_lbft3;
 %  Properties      [       SENSOR          RANGE              ACCURACY       REPEATIBILITY     TEMPERATURE RANGE      TEST
PRESSURE     MAX OP. PRESSURE      WEIGHT        DIMENSIONS - in         POWER     CURRENT     VIBRATION      TOLERANCE
VISCOSITY      FLOWRATES    OPERATIONAL   ]
 %    Units         [        TYPE      Low  lb/ft^3  High      lb/ft^3         lb/ft^3          Low   deg F  High           psi
psi             lb        Len      Wid      Hgt      V DC       mA            g's           lb/ft^3          cP             gal/hr
Type       ]
    DENSITYSENSOR_DB =  { 'MMS 7826'    0          187.284     0.06243         0.06243         -58.0          392.0            0
3002.28           0          0       0       0         25          42            0               0            500              0
'LIQUID'    ;
                        'MMS 7826L'   0          187.284     0.06243         0.06243         -58.0          392.0            0
1450.38           0          0       0       0         28          45            0               0            500              0
'LIQUID'    ;
                        'MMS 7828'    0          187.284     0.06243         0.06243         -58.0          392.0            0
3002.28           0          0       0       0         28          45            0               0           20000             0
'LIQUID'    ;
                        'MMS 7828L'   0          187.284     0.06243         0.06243         -58.0          392.0            0
1450.38           0          0       0       0         28          45            0               0           20000             0
'LIQUID'    ;
                        'MMS 7835'    0          187.284     0.009364        0.0012485       -58.0          230.0
3263.35           2175.57          48.50     40.43     6.73      4.49      28          80            0.6          0.00181
0           792.516       'LIQUID'    ;
                        'MMA 7835'    0          187.284     0.009364        0.0012485       -58.0          230.0
3263.35           2175.57          48.50     40.43     6.73      4.49      28          80            0.6          0.00181
0           792.516       'LIQUID'    ;
                        'EGL 7845'    0          187.284     0.31214           0             -58.0          320.0
2175.57           1450.38          48.50     40.43     6.73      4.49      28          80            0            0.03745
0           792.516       'LIQUID'    ;
                        'EGL 7847'    0          187.284     0.31214           0             -58.0          320.0
435.113           290.075          48.50     40.43     6.73      4.49      28          80            0            0.03745
0           792.516       'LIQUID'    ;
                        'MMT 7845'    0          187.284     0.02185         0.0031214       -58.0          230.0
2175.57           1450.38          61.73     40.43     6.73      4.49      28          17            0            0.03745
0           792.516        'GAS'      ;
                        'MMT 7846'    0          187.284     0.031214        0.0031214       -58.0          230.0
1087.78           725.189          61.73     40.43     6.73      4.49      28          17            0            0.03745
0           792.516        'GAS'      ;
```

```
                        'MMT 7847'    0          187.284     0.02185        0.0031214       -58.0          230.0
435.113           290.075          61.73      40.43    6.73    4.49      28          17            0           0.03745
0            792.516       'GAS'      ;
                        'GDT 7812'   0.06243     249.712     0.06243         0.124856        -4.0          185.0            0
2175.57          11.02       14.33    5.47    5.47       33          25            0            0.03745            0
57.0612       'GAS'       ;
                        '167 LP'     6.2428       99.885     0.93641           0            -320.8          752.0            0
2349.61          40.79       14.25   10.63    8.27        0           0            2.0              0              0
317.006       'LIQUID'   };

    DensityUnitLabelStr = { 'N/A'  'lb/ft^3'  'lb/ft^3'  'lb/ft^3'  'lb/ft^3'   'deg F'  'deg F'   'psi'  'psi'    'lb'   'in'
'in'  'in'  'V-DC'  'mA'    'g''s'   'lb/ft^3'    'cP'   'gal/hr'   'Type'   };

end

DENSITYSENSOR_DB{1,21} = 'continuous real time measurement of fluid density in pipelines, open or closed tanks';
DENSITYSENSOR_DB{2,21} = 'possible of top entry only';
DENSITYSENSOR_DB{3,21} = 'continuous real time measurement of fluid density in pipelines, open or closed tanks';
DENSITYSENSOR_DB{4,21} = 'possible of top entry only';
DENSITYSENSOR_DB{5,21} = 'metering of crude and refined hydrocarbons or non-aggressive process liquids - in pipeline operating
conditions';
DENSITYSENSOR_DB{6,21} = '';
DENSITYSENSOR_DB{7,21} = 'General process, Measurement of liquids with up to 100% entrained gas';
DENSITYSENSOR_DB{8,21} = 'Hygienic, Measurement of liquids with up to 100% entrained gas';
DENSITYSENSOR_DB{9,21} = 'general process';
DENSITYSENSOR_DB{10,21} = 'corrosive applications';
DENSITYSENSOR_DB{11,21} = 'hygienic applications';
DENSITYSENSOR_DB{12,21} = 'Gas blending & Direct measurement of ethylene density - Process Gas Must be dry';
DENSITYSENSOR_DB{13,21} = 'Measuring of liquid level, interface or density with displacer (Archimedes principle) and torque
tube as transmitting element';

%   Sort Database Sensors by GAS or Liquid Function Capability
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


if exist('D_MATRIX2') == 0
    fprintf('\n\n  ***  WARNING  ***');
    fprintf('\n  NO VIABLE DENSITY SENSOR EXISTS IN DATABASE!');
    fprintf('\n  Other methods to determine Planetary Atmospheric Density must be used.\n\n');
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

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,d} = 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('\n\n   ***   D E N S I T Y   S E N S O R   S U M M A R Y   R E S U L T S   ***');
    fprintf('\n\n***   I N P U T S   ***');
    fprintf('\n  Density Sensor Operating Type:      %s', DENS_TYPE);
    fprintf('\n  Planetary Surface Density:   %7.2f  %s', PLDENSITY, DensityUnit);
    fprintf('\n  Unit System:           %s', UNITS);

    fprintf('\n\n*************************************************************************\n');

    %  Density Sensor Properties
    fprintf('\n\n** Sensor Properties **');
    fprintf('\n  Density Sensor Type:              %s', D_MATRIX6{1,1});
    fprintf('\n  Density Sensing Range - Low:   %9.3f   %s', D_MATRIX6{1,2}, DensityUnitLabelStr{2} );
    fprintf('\n                         High:   %9.3f   %s', D_MATRIX6{1,3}, DensityUnitLabelStr{3} );
    fprintf('\n  Sensor Accuracy:               %9.6f   %s', D_MATRIX6{1,4}, DensityUnitLabelStr{4} );
    fprintf('\n  Sensor Repeatability:         %10.7f   %s', D_MATRIX6{1,5}, DensityUnitLabelStr{5} );
    fprintf('\n  Sensor Tolerance:             %10.6f   %s', D_MATRIX6{1,17}, DensityUnitLabelStr{17} );
    fprintf('\n  Sensor Viscosity:              %9.3f   %s', D_MATRIX6{1,18}, DensityUnitLabelStr{18} );
    fprintf('\n  Sensing Flowrates:             %9.3f   %s', D_MATRIX6{1,19}, DensityUnitLabelStr{19} );

    %  Environmental Properties
    fprintf('\n\n   Environmental Properties');
    fprintf('\n  Temperature Range - Low:      %9.3f   %s', D_MATRIX6{1,6}, DensityUnitLabelStr{6} );
    fprintf('\n                       High:   %9.3f   %s', D_MATRIX6{1,7}, DensityUnitLabelStr{7} );
    fprintf('\n  Test Pressure:               %9.3f   %s', D_MATRIX6{1,8}, DensityUnitLabelStr{8} );
    fprintf('\n  Max Operating Pressure:      %9.3f   %s', D_MATRIX6{1,9}, DensityUnitLabelStr{9} );
    fprintf('\n  Vibration Loads:             %9.3f   %s', D_MATRIX6{1,16}, DensityUnitLabelStr{16} );

    fprintf('\n\n  Physical Properties');
    fprintf('\n  Sensor Mass:                  %9.3f   %s', D_MATRIX6{1,10}, DensityUnitLabelStr{10} );
    fprintf('\n  Sensor Dimensions - Length:   %9.3f   %s', D_MATRIX6{1,11}, DensityUnitLabelStr{11} );
    fprintf('\n                      Width:    %9.3f   %s', D_MATRIX6{1,12}, DensityUnitLabelStr{12} );
    fprintf('\n                     Height:    %9.3f   %s', D_MATRIX6{1,13}, DensityUnitLabelStr{13} );

    fprintf('\n\n  Power Requirements');
    fprintf('\n  Input Voltage:                %9.3f   %s', D_MATRIX6{1,14}, DensityUnitLabelStr{14} );
    fprintf('\n  Input Amperage:               %9.3f   %s', D_MATRIX6{1,15}, DensityUnitLabelStr{15} );
```

```
    fprintf('\n\n  Sensor Comments:');
    fprintf('\n  %s', D_MATRIX6{1,21});

end

%************************************************************************
%***     Writes a text file summary of the Planetary Properties

    Fout = fopen('DENSITY_SENSOR_Summary.txt','w+');

    fprintf(Fout, '\n\n   ***   D E N S I T Y   S E N S O R   S U M M A R Y   R E S U L T S   ***');
    fprintf(Fout, '\n\n***   I N P U T S   ***');
    fprintf(Fout, '\n  Density Sensor Operating Type:      %s', DENS_TYPE);
    fprintf(Fout, '\n  Planetary Surface Density:   %7.2f  %s', PLDENSITY, DensityUnit);
    fprintf(Fout, '\n  Unit System:           %s', UNITS);

    fprintf(Fout, '\n\n************************************************************************\n');

    %  Density Sensor Properties
    fprintf(Fout, '\n\n** Sensor Properties **');
    fprintf(Fout, '\n  Density Sensor Type:              %s', D_MATRIX6{1,1});
    fprintf(Fout, '\n  Density Sensing Range - Low:   %9.3f   %s', D_MATRIX6{1,2}, DensityUnitLabelStr{2} );
    fprintf(Fout, '\n                         High:   %9.3f   %s', D_MATRIX6{1,3}, DensityUnitLabelStr{3} );
    fprintf(Fout, '\n  Sensor Accuracy:                %9.6f   %s', D_MATRIX6{1,4}, DensityUnitLabelStr{4} );
    fprintf(Fout, '\n  Sensor Repeatability:          %10.7f   %s', D_MATRIX6{1,5}, DensityUnitLabelStr{5} );
    fprintf(Fout, '\n  Sensor Tolerance:              %10.6f   %s', D_MATRIX6{1,17}, DensityUnitLabelStr{17} );
    fprintf(Fout, '\n  Sensor Viscosity:               %9.3f   %s', D_MATRIX6{1,18}, DensityUnitLabelStr{18} );
    fprintf(Fout, '\n  Sensing Flowrates:              %9.3f   %s', D_MATRIX6{1,19}, DensityUnitLabelStr{19} );

    %  Environmental Properties
    fprintf(Fout, '\n\n   Environmental Properties');
    fprintf(Fout, '\n  Temperature Range - Low:      %9.3f   %s', D_MATRIX6{1,6}, DensityUnitLabelStr{6} );
    fprintf(Fout, '\n                       High:      %9.3f   %s', D_MATRIX6{1,7}, DensityUnitLabelStr{7} );
    fprintf(Fout, '\n  Test Pressure:               %9.3f   %s', D_MATRIX6{1,8}, DensityUnitLabelStr{8} );
    fprintf(Fout, '\n  Max Operating Pressure:      %9.3f   %s', D_MATRIX6{1,9}, DensityUnitLabelStr{9} );
    fprintf(Fout, '\n  Vibration Loads:             %9.3f   %s', D_MATRIX6{1,16}, DensityUnitLabelStr{16} );

    fprintf(Fout, '\n\n  Physical Properties');
    fprintf(Fout, '\n  Sensor Mass:                  %9.3f   %s', D_MATRIX6{1,10}, DensityUnitLabelStr{10} );
    fprintf(Fout, '\n  Sensor Dimensions - Length:   %9.3f   %s', D_MATRIX6{1,11}, DensityUnitLabelStr{11} );
    fprintf(Fout, '\n                       Width:    %9.3f   %s', D_MATRIX6{1,12}, DensityUnitLabelStr{12} );
    fprintf(Fout, '\n                      Height:    %9.3f   %s', D_MATRIX6{1,13}, DensityUnitLabelStr{13} );

    fprintf(Fout, '\n\n  Power Requirements');
    fprintf(Fout, '\n  Input Voltage:                %9.3f   %s', D_MATRIX6{1,14}, DensityUnitLabelStr{14} );
    fprintf(Fout, '\n  Input Amperage:               %9.3f   %s', D_MATRIX6{1,15}, DensityUnitLabelStr{15} );

    fprintf(Fout, '\n\n  Sensor Comments:');
    fprintf(Fout, '\n  %s', D_MATRIX6{1,21});

    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&gclid=CMCQxI2pzpACFRpdagodVmMrXg
%  http://www.spectrometer.com/
%  http://www.thinksrs.com/products/RGA.htm
%  http://www.netzsch-thermal-analysis.com/en/products/detail/pid,33.html
%  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 un Unit System
       %%  2 - D Matrix  GCMSSENSOR_DB(DS_Type, Property)

       if strcmpi(UNITS,'SI') == 1
     %  Properties      [    SENSOR     WAVELENGTH RANGE     SPECTRAL WIDTH   RESOLUTION     SCAN TIME    DETECTOR ARRAY
WEIGHT      DIMENSIONS - mm             POWER               TYPE       SAMPLE SIZE             TYPE               ]
     %    Units       [      TYPE       Low   nm   High             nm               nm          sec         # of Elements
kg        Len      Wid      Hgt           Volts             N/A          microL                N/A                  ]
       GCMSSENSOR_DB =  { 'U-0080D'    190          1100           3              0           0.05          1024
10       420      294     120       [ 100 120 230 240 ]     'AC'    [ 1 2 5 10 20 50 ]    'Photodiode Array'         ;
                          'U-1900'     190          1100           4              0           15.17            0
27       643      380     311     [ 100 120 220 230 240 ]   'AC'        [  0  ]          'Silicon photodiode'      ;
                          'U-2900'     190          1100           1.5            0           15.17            0
31       605      500     283     [ 100 120 220 230 240 ]   'AC'      [ 5 25 50 ]         'Silicon photodiode'      ;
                          'U-3010'     190           900           5              0           23.66            1
50       680      690     330     [ 100 120 220 230 240 ]   'AC'         [ 0 ]         'Single Photomultiplier'     ;
                          'U-4100'     175          2600           8              0           60.63            1
120       800      730     880     [ 100 120 220 230 240 ]   'AC'         [ 0 ]      'Photomultiplier(UV-Vis region) Cooled
PBs(NIR region)' ;
```

```
                        'F-2500'     220          730          20              0          10.20          1
35       630     500    270    [ 100 120 220 230 240 ]   'AC'          [ 0 ]    'Fluorescence Spectrophotometer'   ;
                        'F-7000'     200          750          20              0          0.55           1
41       520     620    300    [ 100 120 220 230 240 ]   'AC'        [ 6000 ]   'Fluorescence Spectrophotometer'   ;
                        'Z-2000'     190          900          2.6             0          0              1
189       630    1100   700         [ 220 230 240 ]          'AC'          [ 100 ]   'Photomultiplier Flame/Graphite Furnace' ;
                        'Z-2300'     190          900          2.6             0          0              1
106       630     750   700       [ 115 220 230 240 ]        'AC'           [ 0 ]          'Photomultiplier Flame'      ;
                        'Z-2700'     190          900          2.6             0          0              1
139       630     750   700         [ 220 230 240 ]          'AC'          [ 100 ]   'Photomultiplier Graphite Furnace' ;
                        'H10'        200          1600         4               1          0              1
1       128     114     84              [ 0 ]           'N/A'          [ 0 ]          'Monochromator'          ;
                        'CP140'      190          2500         0               0          0              2048
0         0       0     0               [ 0 ]           'N/A'          [ 0 ]          'Imaging Spectrograph'     ;
                        'VS140'      190          1100         0               1.4        0.01           3864
0.6       178     123    58             [ 7.5 ]           'DC'          [ 0 ]    'Linear CCD 3864 pixels, Photo Diode Array
1024 pixels'    ;
                        'MicroHR'    150          1500         0               0.3        3              3648
5.0       178     267   140             [  0  ]          'N/A'          [ 0 ]    '2D CCD 3684 pixels, Photo Diode Array 1024
pixels'   ;
                        'TRIAX 180'  185          4000         0               0.3        0              1
0       260     230    320            [  0  ]          'N/A'          [ 0 ]     'Imaging Monochromator/Spectrograph'   ;
                        'H-20'       100          3200         0.5             0.5        0              1
3.2       339     84    84             [  0  ]          'N/A'          [ 0 ]      'High Luminosity Monochromator'   ;
                        'iHR320'     150          4000         0               0.06       24.06          1
20        417     422   192            [  0  ]          'N/A'          [ 0 ]          'Imaging Spectrometer'       ;
                        'iHR550'     150          4000         0               0.025      24.06          1
28        648     460   193            [  0  ]          'N/A'          [ 0 ]          'Imaging Spectrometer'       ;
                        'FHR 640'    185          1500         0               0.016      5.0            1
65        740     350   350            [  0  ]          'N/A'          [ 0 ]        'Monochromator/Spectrograph'    ;
                        '750M'       185          4000         0               0.01       0              1
0       838     311    330            [  0  ]          'N/A'          [ 0 ]       'High resolution spectrometer'   };

       GCMSUnitLabelStr = { 'N/A'  'nm'  'nm'  'nm'  'nm'  'sec'   'N/A'  'kg'   'mm'  'mm'    'mm'  'V'  'N/A'    'microL'
'N/A'  };

       end


       if strcmpi(UNITS,'British') == 1
     %  Properties       [    SENSOR     WAVELENGTH RANGE      SPECTRAL WIDTH    RESOLUTION     SCAN TIME    DETECTOR ARRAY
WEIGHT      DIMENSIONS - in             VOLTAGE              TYPE       SAMPLE SIZE             TYPE                      ]
     %    Units          [     TYPE      Low 10^-7 in High       10^-7 in        10^-9 in        sec        # of Elements
lbm       Len    Wid      Hgt           Volts                N/A          microL                 N/A                       ]
       GCMSSENSOR_DB =  { 'U-0080D'     74.80       433.07          1.1811              0          0.05           1024
22.05     16.54   11.57    4.72     [ 100 120 230 240 ]     'AC'     [ 1 2 5 10 20 50 ]    'Photodiode Array'             ;
                          'U-1900'     74.80       433.07          1.5748              0          15.17          0
59.5      22.5    15       12.3     [ 100 120 220 230 240 ]   'AC'           [ 0 ]          'Silicon photodiode'           ;
                          'U-2900'     74.80       433.07          0.5906              0          15.17          0
68        24      20       11.5     [ 100 120 220 230 240 ]   'AC'         [ 5 25 50 ]      'Silicon photodiode'           ;
                          'U-3010'     74.80       354.33          1.9685              0          23.66          1
110        26.8    27.2    13      [ 100 120 220 230 240 ]   'AC'           [ 0 ]          'Single Photomultiplier'        ;
```

```
                        'U-4100'     68.89      1023.62        3.1496            0          60.63          1
264        31.50   28.74    34.65  [ 100 120 220 230 240 ]   'AC'          [ 0 ]    'Photomultiplier(UV-Vis region) Cooled
PBs(NIR region)' ;
                        'F-2500'     86.61      287.40         7.8740            0          10.20          1
77         24.8    19.7     10.6   [ 100 120 220 230 240 ]   'AC'          [ 0 ]    'Fluorescence Spectrophotometer'     ;
                        'F-7000'     78.74      295.27         7.8740            0          0.55           1
90.39      20.47   24.41    11.81  [ 100 120 220 230 240 ]   'AC'        [ 6000 ]   'Fluorescence Spectrophotometer'     ;
                        'Z-2000'     74.80      354.33         1.0236            0          0              1
416.67     24.80   43.31    27.56      [ 220 230 240 ]       'AC'          [ 100 ]  'Photomultiplier Flame/Graphite Furnace' ;
                        'Z-2300'     74.80      354.33         1.0236            0          0              1
233.70     24.80   29.53    27.56    [ 115 220 230 240 ]     'AC'          [ 0 ]        'Photomultiplier Flame'          ;
                        'Z-2700'     74.80      354.33         1.0236            0          0              1
306.44     24.80   29.53    27.56      [ 220 230 240 ]       'AC'          [ 100 ]    'Photomultiplier Graphite Furnace'  ;
                         'H10'       78.74      629.92         1.5748            39.37      0              1
2.20       5.04    4.49     3.31          [ 0 ]           'N/A'          [ 0 ]            'Monochromator'              ;
                        'CP140'      74.80      984.25         0                 0          0              2048
0          0       0        0             [ 0 ]           'N/A'          [ 0 ]          'Imaging Spectrograph'         ;
                        'VS140'      74.80      433.07         0                 55.12      0.01           3864
1.5        7.0     4.73     2.25          [ 7.5 ]         'DC'           [ 0 ]    'Linear CCD 3864 pixels, Photo Diode Array
1024 pixels'    ;
                        'MicroHR'    59.05      590.55         0                 11.81      3              3648
11.0       7.0     10.5     5.5           [  0  ]          'N/A'          [ 0 ]    '2D CCD 3684 pixels, Photo Diode Array 1024
pixels'   ;
                        'TRIAX 180'  72.83      1574.80        0                 11.81      0              1
0          10.24   9.06     12.60         [  0  ]         'N/A'          [ 0 ]   'Imaging Monochromator/Spectrograph'  ;
                        'H-20'       39.37      1258.94        0.1969            19.69      0              1
7.05       13.35   3.31     3.31          [  0  ]         'N/A'          [ 0 ]     'High Luminosity Monochromator'    ;
                        'iHR320'     59.05      1574.80        0                 2.362      24.06          1
45         16.4    16.6     7.6           [  0  ]         'N/A'          [ 0 ]          'Imaging Spectrometer'          ;
                        'iHR550'     59.05      1574.80        0                 0.9843     24.06          1
62         25.51   18.09    7.78          [  0  ]         'N/A'          [ 0 ]          'Imaging Spectrometer'          ;
                        'FHR 640'    72.83      590.55         0                 0.6299     5.0            1
143.3      29.13   13.78    13.78           [  0  ]         'N/A'          [ 0 ]       'Monochromator/Spectrograph'       ;
                        '750M'       72.83      1574.80        0                 0.3937     0              1
0          32.99   12.24    12.99         [  0  ]         'N/A'          [ 0 ]     'High resolution spectrometer'     };

       GCMSUnitLabelStr = { 'N/A'  '10^-7 in'  '10^-7 in'  '10^-7 in'  '10^-9 in'  'sec'   'N/A'  'lbm'   'in'  'in'    'in'
'V'  'N/A'    'microL'    'N/A'  };

       end


       % Comments on Features / Use
       GCMSSENSOR_DB{1,16} = { 'DNA/RNA measurement'                    'A260/280 ratio, Purity, Concentration, Molar
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'  };
      GCMSSENSOR_DB{2,16} = { 'Modes of operation'  'Scanning, Time (Kinetics), Photometry and Ratio (DNA/RNA Analysis)'  };
      GCMSSENSOR_DB{3,16} = { 'Resolution'  'Satisfies European Pharomacopoeia Bandpass'  };
      GCMSSENSOR_DB{4,16} = { '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{5,16} = { 'Band pass'                       'NIR Region: Switching in 0.2 nm steps from 0.24 to 8.0 nm '
;
                        'Scan speed'                        '0.3 (0.75), 3 (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';
                        'Sample Compartment Dimensions'   '200 x 200 mm Standard Size,  430 x 430 mm Large Sample Size (800
x 930 x 980 mm Large Unit Dims)'  ;
                        'Monochromator system'              'prism/grating or grating/grating'  };
      GCMSSENSOR_DB{6,16} = { 'Band pass'                       'Variable: 2.5, 5, 10 and 20 nm Excitation and Emission
monochromators'  ;
                        'Scan speed '                       '15, 60, 300, 1,500 and 3,000 nm/min';
                        'Data Processing functions'         'photometry, wavelength scanning and time scanning'  };
      GCMSSENSOR_DB{7,16} = { 'Band pass'                       'Excitation side: 1, 2.5, 5, 10, 20 nm  Emission side: 1,
2.5, 5, 10, 20 nm '  ;
                        'Scan speed '                       '30, 60, 240, 1200, 2400, 12000, 30000, and 60000 nm/min';
                        'Response'                          'Response from 0 to 98%  0.002, 0.004, 0.01, 0.05, 0.1, 2, 4 s'
;
                        'Applications'                      'industrial, environmental, pharmaceutical, biotechnology'  };
      GCMSSENSOR_DB{8,16} = { 'Measurement Mode'                'Atomic Absorption and Flame Photometry'  ;
                        'Optics '                           'Double-beam Method (Polarized Zeeman Method)';
                        'Slit width'                        '4 steps (0.2, 0.4, 1.3, 2.6 nm)'  ;
                        'Sample Volume'                     '1 to 100 microL'  ;
                        'Sample Number'                     '60 pcs (1.5 mL container), 96-well microplate [ option ]'  };
      GCMSSENSOR_DB{9,16} = { 'Measurement Mode'                'Atomic Absorption and Flame Photometry'  ;
                        'Optics '                           'Double-beam Method (Polarized Zeeman Method)'  ;
                        'Slit width'                        '4 steps (0.2, 0.4, 1.3, 2.6 nm)'  };
      GCMSSENSOR_DB{10,16} = { 'Measurement Mode'               'Atomic Absorption Spectrometry'  ;
                         'Optics '                          'Double-beam Method (Polarized Zeeman Method)'  ;
                         'Slit width'                       '4 steps (0.2, 0.4, 1.3, 2.6 nm)'  ;
                         'Sample Volume'                    '1 to 100 microL'  ;
                         'Sample Number'                    '60 pcs (1.5 mL container), 96-well microplate [ option ]'  };
      GCMSSENSOR_DB{11,16} = { 'Model Ranges'                   'UV-Range 200 - 750 nm,  Vis-Range 350 - 800 nm,  IR-Range
400 - 1600'  ;
                         'Optical Configuration'            'Aberration-corrected concave holographic diffraction grating'
;
                         'Slit width'                       'Fixed width, interchangeable. 3 pairs supplied: 0.5, 1 and 2 mm'
;
                         'Application'                      'Low resolution absorption transmission and fluorescence'  };
      GCMSSENSOR_DB{12,16} = { 'Models'                         'Multiple Models ranges from 190 to 2500 nm'  ;
                         'Detector Array'                   'CCD - Linear 2D array <2048 pixels,  PDA - NMOS & CMOS 128-1024
pixels'   ;
                         'Optical Configuration'            'Aberration-corrected concave holographic diffraction grating'
;
```

```
                            'Slit width'                    'Fixed width, interchangeable  0.01-1 mm'  ;
                            'Interface'                     'Multiple connection types - ADC, RS232, RS485, USB, Parallel'
};
        GCMSSENSOR_DB{13,16} = { 'Models'                       'UV-Vis 190-800nm, Vis 380-780 nm, Vis-NIR 400-1100 nm'  ;
                            'Detector Array'                'CCD - Linear array 3864 pixels,  PDA - 1024 pixels'   ;
                            'Optical Configuration'         'Aberration-corrected concave holographic diffraction grating'  ;
                            'Data Handling'                 'Real-Time data smoothing and transfer to Excel'  };
        GCMSSENSOR_DB{14,16} = { 'Models'                       'Sygnature CCD arrays for UV-NIR Range,  Linear InGaAs Array
NIR Region'  ;
                            'Detector Array'                'CCD - 2D array 3864 pixels,  PDA - 1024 pixels'   ;
                            'Optical Configuration'         'Aberration-corrected concave holographic diffraction grating'  ;
                            'Interface'                     'USB 2.0 (1.1 Compatible)'  ;
                            'Cooling Options'               'Thermoelectric (TE) 200K,  Liquid Nitrogen (LN) 140K'   };
        GCMSSENSOR_DB{15,16} = { 'Models'                       '180 Single Input-Single Output, 190 Single Input-Dual Output
CCD and IR'  ;
                            'Optical Configuration'         'Corrected Cross Czerny Turner'  ;
                            'Interface'                     'RS-232 and IEEE'  ;
                            'Slit Width'                    'Motorized: 0-2 mm in 2 µm steps or Motorized: 0-7 mm in 6.35 µm
steps or Interchangeable fixed slits'   };
        GCMSSENSOR_DB{16,16} = { 'Models'                       'FUV 100-500nm, UV 200-750nm, Vis 350-800nm, IR 400-1600nm,
FIR 800-3200mn'  ;
                            'Optical Configuration'         'Aberration-corrected concave holographic diffraction grating'  ;
                            'Interface'                     'RS-232 and IEEE488'  ;
                            'Resolution'                    '0.5nm FUV-Vis Region,  1nm IR Region, 2nm FIR Region'   ;
                            'Slit Width'                    'Interchangeable fixed slits 0.5 mm, 1 mm and 2 mm wide'   };
        GCMSSENSOR_DB{17,16} = { 'Features'                     '2 Input, 2 Output Ports'  ;
                            'Interface'                     'USB 2.0 (1.1 Compatible)'  ;
                            'Cooling Options'               'Thermoelectric (TE) 200K,  Liquid Nitrogen (LN) 140K'   ;
                            'Slit Width'                    '0 to 2 mm in 2 ?m steps, 0 to 7 mm in 6.25 ?m steps'   ;
                            'Applications'                  'Raman Spectroscopy, Photoluminescence (PL), Absorption /
Transmission / Reflectance, Fluorescence, Plasma / Emission Analysis'};
        GCMSSENSOR_DB{18,16} = { 'Features'                     '2 Input, 2 Output Ports'  ;
                            'Interface'                     'USB 2.0 (1.1 Compatible)'  ;
                            'Cooling Options'               'Thermoelectric (TE) 200K,  Liquid Nitrogen (LN) 140K'   ;
                            'Slit Width'                    '0 to 2 mm in 2 ?m steps, 0 to 7 mm in 6.25 ?m steps'   ;
                            'Applications'                  'Raman Spectroscopy, Photoluminescence (PL), Absorption /
Transmission / Reflectance, Fluorescence, Plasma / Emission Analysis'};
        GCMSSENSOR_DB{19,16} = { 'Spectral Range'               'UV to IR'  ;
                            'Scan speed'                    '>300 nm/min';
                            'Slit Width'                    '0 to 2 mm, 0 - 7 mm'   ;
                            'Options'                       'Available in multiple monochromator and spectrograph
configurations'   ;
                            'Features'                      'High Speed, Cast Construction, Fully automated direct grating
drive, Dual field at exit for array detectors'  };
        GCMSSENSOR_DB{20,16} = { 'Models'                       '0-1500nm Mechanical Range, 0-4000nm spectral range with
appropriate gratings'  ;
                            'Slit Width'                    'Manually variable slits. Optional automated side entrance and
exit slits. 0-3 mm with calibration every 2 ?m'   ;
                            'Interface'                     'RS-232 and IEEE488'  ;
                            'Applications'                  'laser diode research and low-temperature photoluminescence
applications'   ;
                            'Features'                      'Four port capability, Compatible with most CCDs, Complete line
of arrays and single channel detectors'  };
```

```
%  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) = 820.14;
             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) = 191.391;   % 10^-7 inches
             Lambdas(k+4) = 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
case 'N2'
    if strcmpi(UNITS,'SI') == 1;
        Lambdas(k) = 399.500;
        Lambdas(k+1) = 404.131;
        Lambdas(k+2) = 444.703;
        Lambdas(k+3) = 463.054;
        Lambdas(k+4) = 500.148;
        Lambdas(k+5) = 500.515;
        Lambdas(k+6) = 566.663;
        Lambdas(k+7) = 567.956;
        Lambdas(k+8) = 575.250;
        Lambdas(k+9) = 594.165;
        Lambdas(k+10) = 648.205;
        k = k+11;
    end
    if strcmpi(UNITS,'British') == 1;
        Lambdas(k) = 157.283;     % 10^-7 inches
        Lambdas(k+1) = 159.107;   % 10^-7 inches
        Lambdas(k+2) = 175.078;   % 10^-7 inches
        Lambdas(k+3) = 182.305;   % 10^-7 inches
        Lambdas(k+4) = 196.909;   % 10^-7 inches
        Lambdas(k+5) = 197.053;   % 10^-7 inches
        Lambdas(k+6) = 223.096;   % 10^-7 inches
        Lambdas(k+7) = 223.605;   % 10^-7 inches
        Lambdas(k+8) = 226.476;   % 10^-7 inches
        Lambdas(k+9) = 233.923;   % 10^-7 inches
        Lambdas(k+10) = 255.199;  % 10^-7 inches
        k = k+11;
    end
case 'O2'
    if strcmpi(UNITS,'SI') == 1;
        Lambdas(k) = 391.196;
        Lambdas(k+1) = 397.326;
        Lambdas(k+2) = 407.587;
        Lambdas(k+3) = 418.979;
        Lambdas(k+4) = 441.491;
        Lambdas(k+5) = 464.914;
        Lambdas(k+6) = 615.818;
        Lambdas(k+7) = 645.598;
        Lambdas(k+8) = 700.223;
        k = k+9;
    end
    if strcmpi(UNITS,'British') == 1;
        Lambdas(k) = 154.014;     % 10^-7 inches
        Lambdas(k+1) = 156.428;   % 10^-7 inches
        Lambdas(k+2) = 160.467;   % 10^-7 inches
        Lambdas(k+3) = 164.952;   % 10^-7 inches
        Lambdas(k+4) = 173.815;   % 10^-7 inches
        Lambdas(k+5) = 183.037;   % 10^-7 inches
        Lambdas(k+6) = 242.448;   % 10^-7 inches
        Lambdas(k+7) = 254.172;   % 10^-7 inches
        Lambdas(k+8) = 275.678;   % 10^-7 inches
        k = k+9;
```

```
    end
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) = 137.795;     % 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) = 587.562;
        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) = 411.370;
        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) = 169.638;   % 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
```

```
        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) = 1950.00;
        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) = 381.890;    % 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
case 'Are'
    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) = 487.986;
        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
```

```
            Lambdas(k+7) = 174.252;    % 10^-7 inches
            Lambdas(k+8) = 178.939;    % 10^-7 inches
            Lambdas(k+9) = 181.479;    % 10^-7 inches
            Lambdas(k+10) = 186.097;    % 10^-7 inches
            Lambdas(k+11) = 192.120;    % 10^-7 inches
            Lambdas(k+12) = 274.230;    % 10^-7 inches
            Lambdas(k+13) = 278.237;    % 10^-7 inches
            Lambdas(k+14) = 281.380;    % 10^-7 inches
            k = k+15;
        end
    case 'K'
        if strcmpi(UNITS,'SI') == 1;
            Lambdas(k) = 360.888;
            Lambdas(k+1) = 404.414;
            Lambdas(k+2) = 464.237;
            Lambdas(k+3) = 496.503;
            Lambdas(k+4) = 533.969;
            Lambdas(k+5) = 535.957;
            Lambdas(k+6) = 580.175;
            Lambdas(k+7) = 583.189;
            Lambdas(k+8) = 691.108;
            Lambdas(k+9) = 693.877;
            Lambdas(k+10) = 696.467;
            k = k+11;
        end
        if strcmpi(UNITS,'British') == 1;
            Lambdas(k) = 142.082;      % 10^-7 inches
            Lambdas(k+1) = 159.218;    % 10^-7 inches
            Lambdas(k+2) = 182.770;    % 10^-7 inches
            Lambdas(k+3) = 195.474;    % 10^-7 inches
            Lambdas(k+4) = 210.224;    % 10^-7 inches
            Lambdas(k+5) = 211.007;    % 10^-7 inches
            Lambdas(k+6) = 228.415;    % 10^-7 inches
            Lambdas(k+7) = 229.602;    % 10^-7 inches
            Lambdas(k+8) = 272.090;    % 10^-7 inches
            Lambdas(k+9) = 273.180;    % 10^-7 inches
            Lambdas(k+10) = 274.200;   % 10^-7 inches
            k = k+11;
        end
    case 'CO'
        if strcmpi(UNITS,'SI') == 1;
            Lambdas(k) = 200.079;
            Lambdas(k+1) = 892.624;
            k = k+2;
        end
        if strcmpi(UNITS,'British') == 1;
            Lambdas(k) =  78.771;     % 10^-7 inches
            Lambdas(k+1) = 351.427;   % 10^-7 inches
            k = k+2;
        end
    otherwise
end
end
```

```
%  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
c = 1;
for g = 1:size(GCMSSENSOR_DB,1)
if (Min_Lambda >= GCMSSENSOR_DB{g,2}) && (Max_Lambda <= GCMSSENSOR_DB{g,3})
    for a = 1:size(GCMSSENSOR_DB,2)
        GCMS_MATRIX{c,a} = GCMSSENSOR_DB{g,a};
    end
    c = c+1;
end
end

%  Down Select a Single Sensor If Multiple Options Exist in the Database
if size(GCMS_MATRIX,1) >1
%  Create Mass / Volume Scaling Factor Optimization
for mv = 1:size(GCMS_MATRIX,1)
    GCMS_SCALE_FAC(mv) = GCMS_MATRIX{mv,8} * GCMS_MATRIX{mv,9} * GCMS_MATRIX{mv,10} * GCMS_MATRIX{mv,11};

    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;
    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_MATRIX,2)
            GCMS_MATRIX2{k,j} = GCMS_MATRIX{c,j};
        end
        k = k + 1;
    end
end

%  Copies GCMS_MATRIX to GCMS_MATRIX2 if all Sensors Eliminated based on Mass Volume Scale Factor
if exist('GCMS_MATRIX2') == 0
    GCMS_MATRIX2 = GCMS_MATRIX;
end


%  Down Select Sensor Based On Smallest Resolution Range
if size(GCMS_MATRIX2,1) >1
    for r = 1:size(GCMS_MATRIX2,1)
        GCMS_RES(r) = GCMS_MATRIX2{r,5};
        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
        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

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('\n\n   ***   G C M S   S E N S O R   S U M M A R Y   R E S U L T S   ***');
fprintf('\n\n***   I N P U T S   ***');
fprintf('\n  Unit System:          %s', UNITS);

fprintf('\n\n*************************************************************************\n');

%  GCMS Sensor Properties
fprintf('\n\n** Sensor Properties **');
fprintf('\n  GCMS Sensor Type:              %s', GCMS_MATRIX3{1,1});
fprintf('\n  Sensing Wavelength Range - Low:   %10.2f   %s', GCMS_MATRIX3{1,2}, GCMSUnitLabelStr{2} );
fprintf('\n                           High:   %10.2f   %s', GCMS_MATRIX3{1,3}, GCMSUnitLabelStr{3} );
fprintf('\n  Spectral Width:                 %10.4f   %s', GCMS_MATRIX3{1,4}, GCMSUnitLabelStr{4} );
```

```
fprintf('\n  Resolution:                          %10.4f   %s', GCMS_MATRIX3{1,5}, GCMSUnitLabelStr{5} );
fprintf('\n  Scan Time:                           %10.2f   %s', GCMS_MATRIX3{1,6}, GCMSUnitLabelStr{6} );
fprintf('\n  Sample Volume:                 ');
for s = 1:length(GCMS_MATRIX3{1,14})
    fprintf('  %3d', GCMS_MATRIX3{1,14}(1,s) );
end
fprintf('   %s', GCMSUnitLabelStr{14} );
fprintf('\n  Detector Array # of Elements:     %10d   %s', GCMS_MATRIX3{1,7}, GCMSUnitLabelStr{7} );
fprintf('\n  Detector Type:     %s', GCMS_MATRIX3{1,15} );


fprintf('\n\n  Power Requirements');
fprintf('\n  Voltage Source:           ');
for v = 1:length(GCMS_MATRIX3{1,12})
    fprintf('  %4d', GCMS_MATRIX3{1,12}(1,v) );
end
fprintf('   %s', GCMSUnitLabelStr{12} );
fprintf('\n  Voltage Type:                            %s   %s', GCMS_MATRIX3{1,13}, GCMSUnitLabelStr{13} );


fprintf('\n\n  Physical Properties');
fprintf('\n  Sensor Mass:                     %10.2f   %s', GCMS_MATRIX3{1,8}, GCMSUnitLabelStr{8} );
fprintf('\n  Sensor Dimensions - Length:      %10.2f   %s', GCMS_MATRIX3{1,9}, GCMSUnitLabelStr{9} );
fprintf('\n                       Width:      %10.2f   %s', GCMS_MATRIX3{1,10}, GCMSUnitLabelStr{10} );
fprintf('\n                      Height:      %10.2f   %s', GCMS_MATRIX3{1,11}, GCMSUnitLabelStr{11} );


fprintf('\n\n  Sensor Comments:');
for n = 1:length(GCMS_MATRIX3{1,16})
    fprintf('\n  %40s          %s', GCMS_MATRIX3{1,16}{n,1}, GCMS_MATRIX3{1,16}{n,2} );
end

end


%************************************************************************
%***     Writes a text file summary of the Planetary Properties

Fout = fopen('GCMS_SENSOR_Summary.txt','w+');

fprintf(Fout, '\n\n   ***   G C M S   S E N S O R   S U M M A R Y   R E S U L T S   ***');
fprintf(Fout, '\n\n***   I N P U T S   ***');
fprintf(Fout, '\n  Unit System:           %s', UNITS);

fprintf(Fout, '\n\n************************************************************************\n');

%  GCMS Sensor Properties
fprintf(Fout, '\n\n** Sensor Properties **');
fprintf(Fout, '\n  GCMS Sensor Type:              %s', GCMS_MATRIX3{1,1});
fprintf(Fout, '\n  Sensing Wavelength Range - Low:   %10.2f   %s', GCMS_MATRIX3{1,2}, GCMSUnitLabelStr{2} );
fprintf(Fout, '\n                             High:   %10.2f   %s', GCMS_MATRIX3{1,3}, GCMSUnitLabelStr{3} );
fprintf(Fout, '\n  Spectral Width:                   %10.4f   %s', GCMS_MATRIX3{1,4}, GCMSUnitLabelStr{4} );
fprintf(Fout, '\n  Resolution:                       %10.4f   %s', GCMS_MATRIX3{1,5}, GCMSUnitLabelStr{5} );
fprintf(Fout, '\n  Scan Time:                        %10.2f   %s', GCMS_MATRIX3{1,6}, GCMSUnitLabelStr{6} );
```

```
        fprintf(Fout, '\n  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, '\n  Detector Array # of Elements:     %10d   %s', GCMS_MATRIX3{1,7}, GCMSUnitLabelStr{7} );
        fprintf(Fout, '\n  Detector Type:     %s', GCMS_MATRIX3{1,15} );


        fprintf(Fout, '\n\n  Power Requirements');
        fprintf(Fout, '\n  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, '\n  Voltage Type:                         %s   %s', GCMS_MATRIX3{1,13}, GCMSUnitLabelStr{13} );


        fprintf(Fout, '\n\n  Physical Properties');
        fprintf(Fout, '\n  Sensor Mass:                       %10.2f   %s', GCMS_MATRIX3{1,8}, GCMSUnitLabelStr{8} );
        fprintf(Fout, '\n  Sensor Dimensions - Length:        %10.2f   %s', GCMS_MATRIX3{1,9}, GCMSUnitLabelStr{9} );
        fprintf(Fout, '\n                           Width:    %10.2f   %s', GCMS_MATRIX3{1,10}, GCMSUnitLabelStr{10} );
        fprintf(Fout, '\n                          Height:    %10.2f   %s', GCMS_MATRIX3{1,11}, GCMSUnitLabelStr{11} );


        fprintf(Fout, '\n\n  Sensor Comments:');
        for n = 1:length(GCMS_MATRIX3{1,16})
            fprintf(Fout, '\n  %40s          %s', GCMS_MATRIX3{1,16}{n,1}, GCMS_MATRIX3{1,16}{n,2} );
        end

        fclose(Fout);

end


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
```

```
%%  2 - D Matrix  GCMSSENSOR_DB(DS_Type, Property)

if strcmpi(UNITS,'SI') == 1
 %  Properties       [    SENSOR          MASS RANGE     MASS FILTER                  DETECTOR                                DYNAMIC
SCAN RANGE     RESOLUTION      SENSITIVITY          MIN PARTIAL PRESSURE       OPERATING PRESSURE     MAX OPERATING TEMP
BAKEOUT TEMP       ]
 %    Units            [     TYPE       Low  amu  High       TYPE                        TYPE                                Orders
of Magnitude      amu              A/Torr                        Torr                        Torr                  deg C
deg C            ]
        GCMSSENSOR_DB1 =  {    'RGA100'      1          100    'Quadrupole'    { 'Faraday Cup'  'Electron Multiplier'  }
6                0.5        [  2.E-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'  }
6                0.5        [  2.E-4   200  ]     [  5.0E-11   5.0E-14 ]       [ 1.0E-4   1.0E-6 ]          70.00
300.00          ;
                            'RGA300'      1          300    'Quadrupole'    { 'Faraday Cup'  'Electron Multiplier'  }
6                0.5        [  2.E-4   200  ]     [  5.0E-11   5.0E-14 ]       [ 1.0E-4   1.0E-6 ]          70.00
300.00          ;
                            'Aeolos'      1          300    'Quadrupole'        {  'Electron Multiplier'  }
6                0.5        [       0       ]     [      1.50E-14      ]      [         0       ]          300.00
300.00          ;
                          'GCMS-Huygens'  2          141    'Quadrupole'        {  'Electron Multiplier'  }
8               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'   'deg C'
'deg C'   };

        %  Properties       [   ION SOURCE        MATERIAL          FILAMENT                    FIELD RANGE            ELECTRON
ENERGY        ION ENERGY       FOCUS VOLTAGE       ELECTRON CURRRENT     ]
        %    Units            [      #                Type                Type                 Low W Degas High          Min Volts Max
Volts          Min Volts Max       Min      mA    Max      ]
          GCMSSENSOR_DB2 = {      1                 'SS304'       '2-Thoriated-Iridium'        1           10            25        105
[ 8  12 ]        0        150      0.0              3.5      ;
                                  1                 'SS304'       '2-Thoriated-Iridium'        1           10            25        105
[ 8  12 ]        0        150      0.0              3.5      ;
                                  1                 'SS304'       '2-Thoriated-Iridium'        1           10            25        105
[ 8  12 ]        0        150      0.0              3.5      ;
                                  1                   'N/A'       '2-Iridium Cathodes'         0            0            70         70
[   0   ]        0          0      0.0               0       ;
                                  5                   'N/A'              'N/A'                 1           16            25         70
[ 8  12 ]        0          0      0.0             0.08     };

        GCMSUnitLabelStr2 = { 'N/A'  'N/A'  'N/A'  'W Degas'   'W Degas'   'V'   'V'   'V'   'V'   'V'   'mA'  'mA'   };

        %  Properties       [      DIMENSIONS - cm          MASS STABILITY        POWER REQ          CURRENT       MASS
DATA RATE     POWER-TYP    POWER-AVE     POWER-MAX    ]
        %    Units            [   Length  Width  Height         amu      min          V      Type          A           kg
bits/sec         W           W            W          ]
          GCMSSENSOR_DB3 = {     46.61  10.42   7.88          0.1      30           24     'DC'          2.5         2.72
28800            0          0            0         ;
```

```
                              46.61  10.42   7.88          0.1     30           24    'DC'          2.5        2.72
28800          0          0          0       ;
                              46.61  10.42   7.88          0.1     30           24    'DC'          2.5        2.72
28800          0          0          0       ;
                                 0      0      0           0.0     00            0    'N/A'          0          0
0            0          0          0       ;
                              47.00  19.80  19.80           0      36           28    'DC'          1.68      17.30
960          28.0       41.0        71.0       };

        GCMSUnitLabelStr3 = { 'cm'  'cm'  'cm'  'amu'  'min'  'V'   'N/A'   'A'   'kg'   'bits/sec'   'W'   'W'   'W'   };
    end

    if strcmpi(UNITS,'British') == 1
        %  Properties        [    SENSOR        MASS RANGE     MASS FILTER                DETECTOR                     DYNAMIC
SCAN RANGE    RESOLUTION       SENSITIVITY         MIN PARTIAL PRESSURE      OPERATING PRESSURE    MAX OPERATING TEMP
BAKEOUT TEMP           ]
        %    Units           [     TYPE      Low  amu  High        TYPE                    TYPE                        Orders
of Magnitude      amu             A/psi                        psi                      psi              deg F
deg F           ]
        GCMSSENSOR_DB1 =  {    'RGA100'     1         100   'Quadrupole'   { 'Faraday Cup'  'Electron Multiplier'  }
6              0.5      [ 3.87E-6  3.87 ]     [  9.67E-13   9.67E-16 ]    [ 1.93E-6  1.93E-8 ]        158.00
572.00          ;
                           'RGA200'     1         200   'Quadrupole'   { 'Faraday Cup'  'Electron Multiplier'  }
6              0.5      [ 3.87E-6  3.87 ]     [  9.67E-13   9.67E-16 ]    [ 1.93E-6  1.93E-8 ]        158.00
572.00          ;
                           'RGA300'     1         300   'Quadrupole'   { 'Faraday Cup'  'Electron Multiplier'  }
6              0.5      [ 3.87E-6  3.87 ]     [  9.67E-13   9.67E-16 ]    [ 1.93E-6  1.93E-8 ]        158.00
572.00          ;
                           'Aeolos'     1         300   'Quadrupole'       {  'Electron Multiplier'  }
6              0.5      [      0       ]     [       2.90E-16        ]    [        0         ]        572.00
572.00          ;
                        'GCMS-Huygens'  2         141   'Quadrupole'       {  'Electron Multiplier'  }
8              1E-6     [      0       ]     [       2.465E-18       ]    [     1.450E-1     ]        212.00
0.00           };

        GCMSUnitLabelStr1 = {  'N/A'   'amu'   'amu'   'N/A'   'N/A'   'N/A'   'amu'   'A/psi'   'psi'   'psi'  'deg F'
'deg F'   };

         %  Properties       [   ION SOURCE       MATERIAL        FILAMENT                 FIELD RANGE          ELECTRON
ENERGY        ION ENERGY       FOCUS VOLTAGE       ELECTRON CURRRENT     ]
         %    Units          [      #               Type                Type                Low W Degas High        Min Volts Max
Volts           Min Volts Max       Min      mA    Max        ]
          GCMSSENSOR_DB2 = {       1              'SS304'      '2-Thoriated-Iridium'      1          10          25        105
[ 8  12 ]        0        150      0.0           3.5      ;
                                   1              'SS304'      '2-Thoriated-Iridium'      1          10          25        105
[ 8  12 ]        0        150      0.0           3.5      ;
                                   1              'SS304'      '2-Thoriated-Iridium'      1          10          25        105
[ 8  12 ]        0        150      0.0           3.5      ;
                                   1               'N/A'       '2-Iridium Cathodes'       0           0          70         70
[   0   ]        0          0      0.0            0       ;
                                   5               'N/A'              'N/A'               1          16          25         70
[ 8  12 ]        0          0      0.0           0.08     };
```

```
          GCMSUnitLabelStr2 = { 'N/A'  'N/A'  'N/A'  'W Degas'   'W Degas'   'V'   'V'   'V'   'V'   'V'   'mA'  'mA'   };


         %  Properties       [      DIMENSIONS - in          MASS STABILITY        POWER REQ         CURRENT       MASS
DATA RATE    POWER-TYP    POWER-AVE     POWER-MAX   ]
         %    Units          [   Length  Width  Height         amu     min          V     Type          A          lbm
bits/sec     BTUs/hr     BTUs/hr       BTUs/hr    ]
          GCMSSENSOR_DB3 = {    18.35   4.10   3.10          0.1     30           24    'DC'          2.5        6.00
28800          0          0             0       ;
                             18.35   4.10   3.10          0.1     30           24    'DC'          2.5        6.00
28800          0          0             0       ;
                             18.35   4.10   3.10          0.1     30           24    'DC'          2.5        6.00
28800          0          0             0       ;
                               0       0      0           0.0     00            0    'N/A'         0           0
0            0          0           0       ;
                             18.50   7.80   7.80           0      36           28    'DC'          1.68       38.14
960         95.63      140.02      242.48     },


          GCMSUnitLabelStr3 = { 'in'  'in'  'in'  'amu'  'min'  'V'   'N/A'   'A'   'lbm'   'bits/sec'   'BTUs/hr'   'BTUs/hr'
'BTUs/hr'   };
    end

    % Comments on Features / Use
    GCMSSENSOR_DB4{1,1} = {  'Gas Chromatagraph 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{2,1} = {  'Gas Chromatagraph 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{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 an 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
```

```
    GCMSSENSOR_DB = cat(2,GCMSSENSOR_DB1,GCMSSENSOR_DB2,GCMSSENSOR_DB3,GCMSSENSOR_DB4);
    GCMSUnitLabelStr = cat(2,GCMSUnitLabelStr1,GCMSUnitLabelStr2,GCMSUnitLabelStr3);


%  Build Up Sensor Database of Sensors whose Range Meets Input Operational Range
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

%  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_TYPE');


%*************************************************************************************************************************
    %
    %   ***   PRINT SUMMARY Output Section for 'Print' = 'Y'

    if strcmpi(Print,'Y') == 1

    fprintf('\n\n   ***   G C M S   S E N S O R   S U M M A R Y   R E S U L T S   ***');
    fprintf('\n\n***   I N P U T S   ***');
    fprintf('\n  GCMS Operational Type:      %s', GCMS_TYPE);
    fprintf('\n  GCMS Operational Range:     %s', GCMS_RANGE);
    fprintf('\n  Unit System:               %s', UNITS);

    fprintf('\n\n*************************************************************************\n');

    %  GCMS Sensor Properties
    fprintf('\n\n** Sensor Properties **');
    fprintf('\n\n** Sensor Properties **');
    fprintf('\n  GCMS Sensor Type:                  %s', GCMS_MATRIX3{1,1});
    fprintf('\n  Sensing Mass/Charge Range - Low:   %10.2f   %s', GCMS_MATRIX3{1,2}, GCMSUnitLabelStr{2} );
    fprintf('\n                             High:   %10.2f   %s', GCMS_MATRIX3{1,3}, GCMSUnitLabelStr{3} );
    fprintf('\n  Mass Filter Type:                  %s', GCMS_MATRIX3{1,4});
    fprintf('\n  Detector Operational Type:    ');
    for r = 1:length(GCMS_MATRIX3{1,5})
        fprintf('    %s', GCMS_MATRIX3{1,5}{r} );
```

```
end
fprintf('\n  Dynamic Mass-Charge Scanning Range:     %10d   %s', GCMS_MATRIX3{1,6}, GCMSUnitLabelStr{6} );
fprintf('\n  Scanning Resolution:                   %10.2f   %s', GCMS_MATRIX3{1,7}, GCMSUnitLabelStr{7} );
fprintf('\n  Scanning Sensitivity:                    ');
for r = 1:length(GCMS_MATRIX3{1,8})
    fprintf('   %5.2f', GCMS_MATRIX3{1,8}(r) );
end
fprintf('   %s', GCMSUnitLabelStr{8} );
fprintf('\n  Minimum Detected Partial Pressure:       ');
for r = 1:length(GCMS_MATRIX3{1,9})
    fprintf('   %5.2f', GCMS_MATRIX3{1,9}(r) );
end
fprintf('   %s', GCMSUnitLabelStr{9} );
fprintf('\n  Operating Pressure:                      ');
for r = 1:length(GCMS_MATRIX3{1,10})
    fprintf('   %5.2f', GCMS_MATRIX3{1,10}(r) );
end
fprintf('   %s', GCMSUnitLabelStr{10} );
fprintf('\n  Maximum Operating Temperature:         %10.2f   %s', GCMS_MATRIX3{1,11}, GCMSUnitLabelStr{11} );
%fprintf('\n  Bakeout Temperature:                    %10.2f   %s', GCMS_MATRIX3{1,12}, GCMSUnitLabelStr{12} );
fprintf('\n  Number of Ion Sources:                 %10d   %s', GCMS_MATRIX3{1,13}, GCMSUnitLabelStr{13} );
%fprintf('\n  Construction Material:                                %s', GCMS_MATRIX3{1,14} );
%fprintf('\n  Filament Material:                                    %s', GCMS_MATRIX3{1,15} );
fprintf('\n  Charge Field Range - Low:               %10.2f   %s', GCMS_MATRIX3{1,16}, GCMSUnitLabelStr{16} );
fprintf('\n                       High:             %10.2f   %s', GCMS_MATRIX3{1,17}, GCMSUnitLabelStr{17} );
fprintf('\n  Electron Source Voltage - Low:         %10.2f   %s', GCMS_MATRIX3{1,18}, GCMSUnitLabelStr{18} );
fprintf('\n                            High:        %10.2f   %s', GCMS_MATRIX3{1,19}, GCMSUnitLabelStr{19} );
fprintf('\n  Ion Energy:                          ');
for r = 1:length(GCMS_MATRIX3{1,20})
    fprintf('   %5.2f', GCMS_MATRIX3{1,20}(r) );
end
fprintf('   %s', GCMSUnitLabelStr{20} );
%fprintf('\n  Focus Voltage - Low:              %10.2f   %s', GCMS_MATRIX3{1,21}, GCMSUnitLabelStr{21} );
%fprintf('\n                  High:             %10.2f   %s', GCMS_MATRIX3{1,22}, GCMSUnitLabelStr{22} );
fprintf('\n  Electron Source Current - Low:         %10 2f   %s', GCMS_MATRIX3{1,23}, GCMSUnitLabelStr{23} );
fprintf('\n                            High:        %10.2f   %s', GCMS_MATRIX3{1,24}, GCMSUnitLabelStr{24} );
fprintf('\n  Warm Up Period Mass Stability:         %10.2f   %s', GCMS_MATRIX3{1,28}, GCMSUnitLabelStr{28} );
fprintf('\n                            Time:        %10.2f   %s', GCMS_MATRIX3{1,29}, GCMSUnitLabelStr{29} );
fprintf('\n  Sensor Return Data Rate:               %10.2f   %s', GCMS_MATRIX3{1,34}, GCMSUnitLabelStr{34} );


fprintf('\n\n  Physical Properties');
fprintf('\n  Sensor Mass:                           %10.2f   %s', GCMS_MATRIX3{1,33}, GCMSUnitLabelStr{33} );
fprintf('\n  Sensor Dimensions - Length:            %10.2f   %s', GCMS_MATRIX3{1,25}, GCMSUnitLabelStr{25} );
fprintf('\n                       Width:            %10.2f   %s', GCMS_MATRIX3{1,26}, GCMSUnitLabelStr{26} );
fprintf('\n                      Height:            %10.2f   %s', GCMS_MATRIX3{1,27}, GCMSUnitLabelStr{27} );
fprintf('\n  Power Source Voltage:                  %10.2f   %s', GCMS_MATRIX3{1,30}, GCMSUnitLabelStr{30} );
fprintf('\n  Power Source Voltage Type:                      %s', GCMS_MATRIX3{1,31} );
fprintf('\n  Power Source Current:                  %10.2f   %s', GCMS_MATRIX3{1,32}, GCMSUnitLabelStr{32} );
fprintf('\n  Power Requirements - Typical:          %10.2f   %s', GCMS_MATRIX3{1,35}, GCMSUnitLabelStr{35} );
fprintf('\n                       Average:          %10.2f   %s', GCMS_MATRIX3{1,36}, GCMSUnitLabelStr{36} );
fprintf('\n                          Peak:          %10.2f   %s', GCMS_MATRIX3{1,37}, GCMSUnitLabelStr{37} );
```

```
fprintf('\n\n  Sensor Comments:');
for n = 1:length(GCMS_MATRIX3{1,38})
    fprintf('\n  %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, '\n\n   ***   G C M S   S E N S O R   S U M M A R Y   R E S U L T S   ***');
fprintf(Fout, '\n\n***   I N P U T S   ***');
fprintf(Fout, '\n  GCMS Operational Type:     %s', GCMS_TYPE);
fprintf(Fout, '\n  GCMS Operational Range:    %s', GCMS_RANGE);
fprintf(Fout, '\n  Unit System:               %s', UNITS);

fprintf(Fout, '\n\n************************************************************************\n');

%  GCMS Sensor Properties
fprintf(Fout, '\n\n** Sensor Properties **');
fprintf(Fout, '\n  GCMS Sensor Type:                    %s', GCMS_MATRIX3{1,1});
fprintf(Fout, '\n  Sensing Mass/Charge Range - Low:   %10.2f   %s', GCMS_MATRIX3{1,2}, GCMSUnitLabelStr{2} );
fprintf(Fout, '\n                             High:   %10.2f   %s', GCMS_MATRIX3{1,3}, GCMSUnitLabelStr{3} );
fprintf(Fout, '\n  Mass Filter Type:                    %s', GCMS_MATRIX3{1,4});
fprintf(Fout, '\n  Detector Operational Type:    ');
for r = 1:length(GCMS_MATRIX3{1,5})
    fprintf(Fout, '    %s', GCMS_MATRIX3{1,5}{r} );
end
fprintf(Fout, '\n  Dynamic Mass-Charge Scanning Range:     %10d   %s', GCMS_MATRIX3{1,6}, GCMSUnitLabelStr{6} );
fprintf(Fout, '\n  Scanning Resolution:                    %10.2f   %s', GCMS_MATRIX3{1,7}, GCMSUnitLabelStr{7} );
fprintf(Fout, '\n  Scanning Sensitivity:                     ');
for r = 1:length(GCMS_MATRIX3{1,8})
    fprintf(Fout, '   %5.2f', GCMS_MATRIX3{1,8}(r) );
end
fprintf(Fout, '   %s', GCMSUnitLabelStr{8} );
fprintf(Fout, '\n  Minimum Detected Partial Pressure:        ');
for r = 1:length(GCMS_MATRIX3{1,9})
    fprintf(Fout, '   %5.2f', GCMS_MATRIX3{1,9}(r) );
end
fprintf(Fout, '   %s', GCMSUnitLabelStr{9} );
fprintf(Fout, '\n  Operating Pressure:                       ');
for r = 1:length(GCMS_MATRIX3{1,10})
    fprintf(Fout, '   %5.2f', GCMS_MATRIX3{1,10}(r) );
end
fprintf(Fout, '   %s', GCMSUnitLabelStr{10} );
fprintf(Fout, '\n  Maximum Operating Temperature:          %10.2f   %s', GCMS_MATRIX3{1,11}, GCMSUnitLabelStr{11} );
%fprintf(Fout, '\n  Bakeout Temperature:                     %10.2f   %s', GCMS_MATRIX3{1,12}, GCMSUnitLabelStr{12} );
fprintf(Fout, '\n  Number of Ion Sources:                  %10d   %s', GCMS_MATRIX3{1,13}, GCMSUnitLabelStr{13} );
%fprintf(Fout, '\n  Construction Material:                             %s', GCMS_MATRIX3{1,14} );
%fprintf(Fout, '\n  Filament Material:                                 %s', GCMS_MATRIX3{1,15} );
fprintf(Fout, '\n  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:                              ');
        for r = 1:length(GCMS_MATRIX3{1,20})
            fprintf(Fout, '   %5.2f', GCMS_MATRIX3{1,20}(r) );
        end
        fprintf(Fout, '   %s', GCMSUnitLabelStr{20} );
        %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{1,28}, GCMSUnitLabelStr{28} );
        fprintf(Fout, '\n                             Time:           %10.2f   %s', GCMS_MATRIX3{1,29}, GCMSUnitLabelStr{29} );
        fprintf(Fout, '\n  Sensor Return Data Rate:                   %10.2f   %s', GCMS_MATRIX3{1,34}, GCMSUnitLabelStr{34} );


        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} );


        fprintf(Fout, '\n\n  Sensor Comments:');
        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         [      Sensor    Humidity Range      Capacitance at 55% RH     Sensitivity    Hysteresis   Response Time
Stability   Temp Range    Frequency Range   Length    Width    Height   Weight ]
 % Units              [        #       Min  %RH  Max     Min pF   Typ pF    Max pF     pF/%RH         +-%RH         sec
%RH/yr   Min deg C Max   Min  kHz  Max       mm         mm       mm       gr     ]
    HUMIDITYSENSOR_DB = {  'HCH-1000'   0         100      310      330      350        0.6             2            15
0.2      -40       120     1        100      18.68     6.52     2.96     0     ;
                           'HIH-3610'   0         100      0        0        0          0             1.2          15
0.2      -40        85     0        0        21.66     5.08     2.03     0     ;
                           'HIH-4000'   0         100      0        0        0          0             3            15
1.2      -40        85     0        0        21.66     5.08     2.03     0     };

    HumUnitLabelStr = { 'sensor' '% RH'  '% RH' 'pF' 'pF' 'pF' 'pF/%RH' '+-%RH' 'sec' '%RH/yr' 'deg C' 'deg C' 'kHz'  'kHz'
'mm'  'mm'  'mm'  'gr' };

end

if strcmpi(UNITS,'BRITISH') == 1
 % Properties         [      Sensor    Humidity Range      Capacitance at 55% RH     Sensitivity    Hysteresis   Response Time
Stability   Temp Range    Frequency Range   Length    Width    Height   Weight ]
 % Units              [        #       Min  %RH  Max     Min pF   Typ pF    Max pF     pF/%RH         +-%RH         sec
%RH/yr   Min deg F Max   Min  kHz  Max       in         in       in       oz     ]
    HUMIDITYSENSOR_DB = {  'HCH-1000'   0         100      310      330      350        0.6             2            15
0.2      -40       248     1        100      0.735     0.257    0.116    0     ;
                           'HIH-3610'   0         100      0        0        0          0             1.2          15
0.2      -40       185     0        0        0.853     0.200    0.080    0     ;
                           'HIH-4000'   0         100      0        0        0          0             3            15
1.2      -40       185     0        0        0.853     0.200    0.080    0     };

    HumUnitLabelStr = { 'N/A' '% RH'  '% RH' 'pF' 'pF' 'pF' 'pF/%RH' '+-%RH' 'sec' '%RH/yr' 'deg F' 'deg F' 'kHz'  'kHz'  'in'
'in'  'in'  'oz' };
```

```
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{k,j} = H_MATRIX{c,j};
            end
        end
        k = k + 1;
    end
    clear H_MATRIX;
    H_MATRIX = H_MATRIX2;

end


% 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;
end

H20_FLG = 0;
% Search Planetary Atmopsphere 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('\n\n  ***   WARNING IN HUMIDITY SENSOR PROGRAM   ***');
        fprintf('\n  Humidity Sensor Not Required In Sensor Package Design!!');
        fprintf('\n  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');


%*************************************************************************************************************************
%
%   ***   PRINT SUMMARY Output Section for 'Print' = 'Y'

if strcmpi(Print,'Y') == 1
```

```
    fprintf('\n\n   ***   H U M I D I T Y   S E N S O R   S U M M A R Y   R E S U L T S   ***');
    fprintf('\n\n***   I N P U T S   ***');
    fprintf('\n  Unit System:          %s', UNITS);

    if H20_FLG == 0;
        fprintf('\n\n  ***   WARNING IN HUMIDITY SENSOR PROGRAM   ***');
        fprintf('\n  Humidity Sensor Not Required In Sensor Package Design!!');
        fprintf('\n  Water Component not found in Selected Planetary Atmosphere.');
    end


    fprintf('\n\n*************************************************************************\n');

    %  Humidity Sensor Properties
    fprintf('\n\n** Sensor Properties **');
    fprintf('\n  Humidity Sensor Type:              %s', H_MATRIX{1,1});
    fprintf('\n  Humidity Sensing Range - Low:   %9.3f   %s', H_MATRIX{1,2}, HumUnitLabelStr{2} );
    fprintf('\n                          High:   %9.3f   %s', H_MATRIX{1,3}, HumUnitLabelStr{3} );
    fprintf('\n  Capacitance at 55%% RH - Low:    %9.3f   %s', H_MATRIX{1,4}, HumUnitLabelStr{4} );
    fprintf('\n                       Typical:   %9.3f   %s', H_MATRIX{1,5}, HumUnitLabelStr{5} );
    fprintf('\n                           Max:   %9.3f   %s', H_MATRIX{1,6}, HumUnitLabelStr{6} );
    fprintf('\n  Sensor Sensitivity:             %9.3f   %s', H_MATRIX{1,7}, HumUnitLabelStr{7} );
    fprintf('\n  Sensor Hysteresis:              %9.3f   %s', H_MATRIX{1,8}, HumUnitLabelStr{8} );
    fprintf('\n  Response Time:                  %9.3f   %s', H_MATRIX{1,9}, HumUnitLabelStr{9} );
    fprintf('\n  Sensor Stability:               %9.3f   %s', H_MATRIX{1,10}, HumUnitLabelStr{10} );

    %  Environmental Properties
    fprintf('\n\n   Environmental Properties');
    fprintf('\n  Temperature Range - Low:        %9.3f   %s', H_MATRIX{1,11}, HumUnitLabelStr{11} );
    fprintf('\n                         High:    %9.3f   %s', H_MATRIX{1,12}, HumUnitLabelStr{12} );
    fprintf('\n  Sensing Frequency Range - Min:  %9.3f   %s', H_MATRIX{1,13}, HumUnitLabelStr{13} );
    fprintf('\n                            Max:  %9.3f   %s', H_MATRIX{1,14}, HumUnitLabelStr{14} );

    fprintf('\n\n  Physical Properties');
    fprintf('\n  Sensor Dimensions - Length:     %9.3f   %s', H_MATRIX{1,15}, HumUnitLabelStr{15} );
    fprintf('\n                       Width:     %9.3f   %s', H_MATRIX{1,16}, HumUnitLabelStr{16} );
    fprintf('\n                      Height:     %9.3f   %s', H_MATRIX{1,17}, HumUnitLabelStr{17} );
    fprintf('\n  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, '\n\n   ***   H U M I D I T Y   S E N S O R   S U M M A R Y   R E S U L T S   ***');
    fprintf(Fout, '\n\n***   I N P U T S   ***');
    fprintf(Fout, '\n  Unit System:          %s', UNITS);

    if H20_FLG == 0;
        fprintf(Fout, '\n\n  ***   WARNING IN HUMIDITY SENSOR PROGRAM   ***');
```

```
    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\n*************************************************************************\n');

%  Humidity Sensor Properties
fprintf(Fout, '\n\n** 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                          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\n   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} );

fprintf(Fout, '\n\n  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.optronics.com/products/commerce.exe?search=action&category=2050&keywords=all&template=Templates/Category.html
%  http://bssc.sel.sony.com/BroadcastandBusiness/docs/brochures/dxc990.pdf
%  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              IMAGING TYPE                 IMAGE                       ARRAY DIMENSIONS
PIXEL SIZE          IMAGE AREA            READ NOISE         FULL WELL CAPACITY       GAIN       LINEARITY      ]
    %    Units           [         NAME                    Type                        TYPE                  LENGTH  # pixels
WIDTH        microm      LENGTH   mm   WIDTH      @ 1MHz  e-  @ 250kHz      Pixel  ke-  Register    e-/ADU         %          ]
    IMAGINGSENSOR_DB1 =  {  'Condor 486:90'              'CAMERA'                  {  'X-Ray' }                4096              4096
13          61.44        61.44       10          5        100          800        1.5         1.0          ;   %1
                            'Condor 486:135'             'CAMERA'                  {  'X-Ray' }                4096              4096
15          92.16        92.16       10          5        100          800        1.5         1.0          ;   %2
                            'Condor 486:200'             'CAMERA'                  {  'X-Ray' }                4096              4096
15         135.10       135.10       10          5        100          800        1.5         1.0          ;   %3
                              'Harrier 447'              'CAMERA'                  { 'VISUAL' }                2048              2048
15          31.00        31.00       12          0         80            0        2.5         1.0          ;   %4
                            'Peregrine 3041'             'CAMERA'                  { 'VISUAL' }                2048              2048
15          30.72        30.72       10          0        100          800        1.4         1.0          ;   %5
                            'Peregrine 486'              'CAMERA'             { 'UV' 'VISUAL' 'NIR' }          4096              4096
15          61.40        61.40       10          5        100          800        1.5         1.0          ;   %6
                            'Peregrine 486SX'            'CAMERA'        { 'X-RAY' 'UV' 'VISUAL' 'NIR' }       4096              4096
15          61.40        61.40       10          5        100          800        1.5         1.0          ;   %7
                            'Macrofire - MC'             'CAMERA'           {   'MICRO' 'VISUAL' }             2048              2048
7.4         15.20        15.20        0          0         40            0          0           0          ;   %8
                        'Macrofire-MC 1600x1200P'        'CAMERA'           {   'MICRO' 'VISUAL' }             1600              1200
7.4         11.84         8.88        0          0         40            0          0           0          ;   %9
                            'QuantiFIRE XI'              'CAMERA'         {  'MICRO' 'VISUAL' 'NIR'  }         2048              2048
7.4         15.20        15.20        0          0         40            0          0           0          ;   %10
```

```
                          'Pixelfly QE'              'CAMERA'          {  'UV' 'VISUAL' 'NIR'  }          1392              1024
6.45          8.98          6.60          0              0          18              0              0          2.0          ;   %11
                          'Sony DXC-990P'            'CAMERA'             { 'VISUAL' }                     752               582
8             6.40          4.80          0              0          0               0              0          0            ;   %12
                            'CCD 3041'                 'ARRAY'    { 'X-RAY' 'UV' 'VISUAL' 'NIR' 'MICRO'}   2048              2048
15            30.72         30.72         7              0          0               750            0          0            ;   %13
                            'CCD 412'                  'ARRAY'    { 'X-RAY' 'UV' 'VISUAL' 'NIR' 'MICRO'}   512               1024
15            15.36         15.36         45             0          85              0              0          3.0          ;   %14
                            'CCD 424'                  'ARRAY'    { 'X-RAY' 'UV' 'VISUAL' 'NIR' 'MICRO'}   1024              1024
21            21.50         21.50         0              0          0               500            0          0            ;   %15
                            'CCD 442A'                 'ARRAY'    { 'X-RAY' 'UV' 'VISUAL' 'NIR' 'MICRO'}   2048              2048
15            30.72         30.72         0              7          0               200            0          0            ;   %16
                            'CCD 447'                  'ARRAY'    { 'X-RAY' 'UV' 'VISUAL' 'NIR' 'MICRO'}   2048              2048
15            30.72         30.72         12             0          80              100            2.5        1.0          ;   %17
                            'CCD 485'                  'ARRAY'    { 'X-RAY' 'UV' 'VISUAL' 'NIR' 'MICRO'}   4096              4096
15            61.20         61.20         12             0          0               750            0          0            ;   %18
                            'CCD 111'                  'LINEAR'   { 'X-RAY' 'UV' 'VISUAL' 'NIR' 'MICRO'}     1                256
13             0.013         3.33         0              0          0               0              0          0            ;   %19
                            'CCD 153A'                 'LINEAR'   { 'X-RAY' 'UV' 'VISUAL' 'NIR' 'MICRO'}     1                512
13             0.013         6.66         0              0          0               0              0          0            };  %20

    ImgUnitLabelStr1 = { 'N/A'  'N/A'  'N/A'  '# Pixels'  '# Pixels'  'micro-m'  'mm'  'mm'  'e-'  'e-'  'ke-'  'ke-'  'e-/ADU'
'%'   };

 %  Properties          [    OPERATING TEMP    COOLING TEMP        COOLING METHOD              ADC DYNAMIC RANGE     READOUT
RATES          READOUT TIME             FRAME RATE              DIMENSIONS  (mm)        WEIGHT     VOLTAGE    V TYPE   CURRENT
POWER   ]
 %    Units             {   Low  deg C  High      deg C                  N/A                          N/A                  MHz
sec                     fps              Length   Width   Height      kg          V        N/A       mA          mW     ]
    IMAGINGSENSOR_DB2 = {   15.0         30.0     -60.0      'Thermoelectric w/Chilled Water'       '16-bit'      [ 4  2.5  2.1
1.6 ]   [ 6.5 1.90 1.18 0.55 ]  [ 0.15 0.52 0.85 1.8 ]      152     145     266        7.7          0        'N/A'        0
0    ;   %1
                        15.0         30.0     -60.0       'Thermoelectric w/Cryo Cooler'        '16-bit'      [ 4  2.5  2.1
1.6 ]   [ 6.5 1.90 1.18 0.55 ]  [ 0.15 0.52 0.85 1.8 ]      165     131     395        16.4         0        'N/A'        0
0    ;   %2
                        15.0         30.0     -60.0       'Thermoelectric w/Cryo Cooler'        '16-bit'      [ 4  2.5  2.1
1.6 ]   [ 6.5 1.90 1.18 0.55 ]  [ 0.15 0.52 0.85 1.8 ]      0       0       0          29.5         0        'N/A'        0
0    ;   %3
                          0            0        0.0                 'N/A'                        '14-bit'           [ 4 ]
[ 0 ]                  [ 0 ]                 0        0       0          0          0       'N/A'       0          0     ;   %4
                        15.0         30.0     -25.0       'Thermoelectric w/Forced Air'         '16-bit'           [ 2 1 ]
[ 1.1 0.364 0.180]       [ 0.9 2.7 5.6 ]        131     112     180        4.5          0       'N/A'       0          0     ;
%5
                        15.0         30.0     -60.0       'Thermoelectric w/Chilled Water'      '16-bit'      [ 4  2.5  2.1
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
0    ;   %6
                        15.0         30.0     -60.0       'Thermoelectric w/Chilled Water'      '16-bit'      [ 4  2.5  2.1
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
0    ;   %7
                          0            0        0.0                 'N/A'                        '12-bit'           [ 0 ]
[ 0 60 ]           [ 5 9 12 15 18 38 ]      134.6   134.6    58.4        0          0       'N/A'       0          0     ;   %8
```

```
                                0           0         0.0                    'N/A'                        '12-bit'              [ 0 ]
[ 0.0005 60 ]          [ 10 18 24 30 48 ]      134.6   134.6    63.5         0          0       'N/A'        0            0    ;
%9
                                0           0       -20.0      'Thermoelectric to -20 Ambient'          '12-bit'              [ 0 ]
[ 0 1200 ]          [ 5 9 12 15 18 38 ]       137.2   137.2    127          0          0       'N/A'        0            0    ;
%10
                              10.0        40.0        0     'Digital Temperature Compensation'         '12-bit'              [ 20 ]
[ 0.000005 65 ]            [ 12 23 ]             39.0    39.0    68.0       0.27         0       'N/A'        0            0    ;
%11
                              -5.0        45.0        0                    'N/A'                        '10-bit'              [ 0.0157
]           [ 0.000001 0.5 ]              [ 0 ]               70.0    72.0   123.5      0.630        0       'N/A'        0
0    ;   %12
                            -100.0        40.0        0                    'N/A'                        'N/A'                 [ 2 1 ]
[ 1.1 0.364 0.180]          [ 0.9 2.7 5.6 ]          45.72   36.83   2.01        0          24        'DC'         0            0    ;
%13
                             -55.0       125.0        0                    'N/A'                        'N/A'                 [ 0 ]
[  0  ]                     [ 0 ]                 0       0       0          0          24        'DC'         5           50    ;
%14
                             -30.0       100.0        0                    'N/A'                        'N/A'                 [ 0 ]
[  0  ]                     [ 0 ]                73.15   52.83   6.096       0          24        'DC'         0            0    ;
%15
                               0           0          0                    'N/A'                        'N/A'                 [ 0 ]
[  0  ]                     [ 0 ]                40.64   40.64   11.05       0          22        'DC'         0            0    ;
%16
                               0           0          0                    'N/A'                        'N/A'                 [ 0 ]
[  0  ]                     [ 0 ]                50.80   50.80    0          0          24        'DC'         0            0    ;
%17
                             -90.0        40.0        0                    'N/A'                        'N/A'                 [ 0 ]
[  0  ]                     [ 0 ]                71.00   71.00    5.08       0          20        'DC'         0            0    ;
%18
                             -25.0        55.0        0                    'N/A'                        'N/A'                 [ 10 ]
[  0  ]                     [ 0 ]                22.86    6.35    7.44       0          15        'DC'         0          100    ;
%19
                             -25.0        70.0        0                    'N/A'                        'N/A'                 [ 20 ]
[  0  ]                     [ 0 ]                30.48   15.24    8.64       0          15        'DC'         0          170    };
%20

    ImgUnitLabelStr2 = { 'deg C'  'deg C'  'deg C'  'N/A'  'N/A'  'MHz'  'sec'  'fps'  'mm'  'mm'  'mm'  'kg'  'V'  'Type'
'mA'   'mW'   };

end


if strcmpi(UNITS,'British') == 1
    %  Properties          [      SENSOR                IMAGING TYPE                    IMAGE                    ARRAY DIMENSIONS
PIXEL SIZE          IMAGE AREA                READ NOISE          FULL WELL CAPACITY       GAIN     LINEARITY     ]
    %    Units            [       NAME                    Type                          TYPE                  LENGTH  # pixels  WIDTH
milli-in     LENGTH   in   WIDTH     @ 1MHz  e-  @ 250kHz    Pixel  ke-  Register    e-/ADU        %          ]
    IMAGINGSENSOR_DB1 =  {  'Condor 486:90'          'CAMERA'                 {  'X-Ray' }              4096                4096
0.5118      2.419        2.419       10            5       100          800          1.5        1.0          ;   %1
                            'Condor 486:135'         'CAMERA'                 {  'X-Ray' }              4096                4096
0.5905      3.628        3.628       10            5       100          800          1.5        1.0          ;   %2
                            'Condor 486:200'         'CAMERA'                 {  'X-Ray' }              4096                4096
0.5905      5.319        5.319       10            5       100          800          1.5        1.0          ;   %3
```

```
                        'Harrier 447'          'CAMERA'              { 'VISUAL' }                2048              2048
0.5905      1.220       1.220       12          0       80          0          2.5        1.0        ;   %4
                        'Peregrine 3041'       'CAMERA'              { 'VISUAL' }                2048              2048
0.5905      1.209       1.209       10          0       100       800          1.4        1.0        ;   %5
                        'Peregrine 486'        'CAMERA'              { 'UV' 'VISUAL' 'NIR' }     4096              4096
0.5905      2.417       2.417       10          5       100       800          1.5        1.0        ;   %6
                        'Peregrine 486SX'      'CAMERA'          { 'X-RAY' 'UV' 'VISUAL' 'NIR' } 4096              4096
0.5905      2.417       2.417       10          5       100       800          1.5        1.0        ;   %7
                        'Macrofire - MC'       'CAMERA'            {   'MICRO' 'VISUAL' }        2048              2048
0.2913      0.598       0.598       0           0       40          0          0          0          ;   %8
                    'Macrofire-MC 1600x1200P'  'CAMERA'            {   'MICRO' 'VISUAL' }        1600              1200
0.2913      0.466       0.350       0           0       40          0          0          0          ;   %9
                        'QuantiFIRE XI'        'CAMERA'          {  'MICRO' 'VISUAL' 'NIR'  }    2048              2048
0.2913      0.598       0.598       0           0       40          0          0          0          ;   %10
                        'Pixelfly QE'          'CAMERA'            {  'UV' 'VISUAL' 'NIR'  }     1392              1024
0.2539      0.353       0.260       0           0       18          0          0          2.0        ;   %11
                        'Sony DXC-990P'        'CAMERA'              { 'VISUAL' }                752               582
0.3150      0.252       0.189       0           0       0           0          0          0          ;   %12
                        'CCD 3041'             'ARRAY'   { 'X-RAY' 'UV' 'VISUAL' 'NIR' 'MICRO'}  2048              2048
0.5905      1.210       1.210       7           0       0         750          0          0          ;   %13
                        'CCD 412'              'ARRAY'   { 'X-RAY' 'UV' 'VISUAL' 'NIR' 'MICRO'}  512               1024
0.5905      0.605       0.605       45          0       85          0          0          3.0        ;   %14
                        'CCD 424'              'ARRAY'   { 'X-RAY' 'UV' 'VISUAL' 'NIR' 'MICRO'}  1024              1024
0.8268      0.846       0.846       0           0       0         500          0          0          ;   %15
                        'CCD 442A'             'ARRAY'   { 'X-RAY' 'UV' 'VISUAL' 'NIR' 'MICRO'}  2048              2048
0.5905      1.209       1.209       0           7       0         200          0          0          ;   %16
                        'CCD 447'              'ARRAY'   { 'X-RAY' 'UV' 'VISUAL' 'NIR' 'MICRO'}  2048              2048
0.5905      1.220       1.220       12          0       80        100          2.5        1.0        ;   %17
                        'CCD 485'              'ARRAY'   { 'X-RAY' 'UV' 'VISUAL' 'NIR' 'MICRO'}  4096              4096
0.5905      2.410       2.410       12          0       0         750          0          0          ;   %18
                        'CCD 111'              'LINEAR'  { 'X-RAY' 'UV' 'VISUAL' 'NIR' 'MICRO'}  1                 256
0.5118      5.118E-03   0.131       0           0       0         0            0          0          ;   %19
                        'CCD 153A'             'LINEAR'  { 'X-RAY' 'UV' 'VISUAL' 'NIR' 'MICRO'}  1                 512
0.5118      5.118E-03   0.262       0           0       0         0            0          0          };  %20

    ImgUnitLabelStr1 = { 'N/A'  'N/A'  'N/A'  '# Pixels'  '# Pixels'  'milli-in'  'in'  'in'  'e-'  'e-'  'ke-'  'ke-'  'e-
/ADU'  '%'  };

 %  Properties          [    OPERATING TEMP    COOLING TEMP        COOLING METHOD              ADC DYNAMIC RANGE     READOUT
RATES          READOUT TIME              FRAME RATE              DIMENSIONS  (in)        WEIGHT    VOLTAGE    V TYPE   CURRENT
POWER   ]
 %    Units              {   Low  deg F  High      deg F                    N/A                              N/A                 MHz
sec                      fps              Length    Width   Height      lbm        V        N/A        mA        BTUs'hr  ]
    IMAGINGSENSOR_DB2 = {   59.0         86.0      -76.0      'Thermoelectric w/Chilled Water'       '16-bit'      [ 4  2.5  2.1
1.6 ]   [ 6.5 1.90 1.18 0.55 ]  [ 0.15 0.52 0.85 1.8 ]     6.00     5.70     10.50      17.0       0       'N/A'       0
0    ;   %1
                         59.0         86.0      -76.0       'Thermoelectric w/Cryo Cooler'        '16-bit'      [ 4  2.5  2.1
1.6 ]   [ 6.5 1.90 1.18 0.55 ]  [ 0.15 0.52 0.85 1.8 ]     6.50     5.20     15.60      36.0       0       'N/A'       0
0    ;   %2
                         59.0         86.0      -76.0       'Thermoelectric w/Cryo Cooler'        '16-bit'      [ 4  2.5  2.1
1.6 ]   [ 6.5 1.90 1.18 0.55 ]  [ 0.15 0.52 0.85 1.8 ]       0        0        0        65.0       0       'N/A'       0
0    ;   %3
```

```
                                0           0           0                   'N/A'                       '14-bit'              [ 4 ]
[ 0 ]                       [ 0 ]               0           0           0           0       'N/A'       0           0     ;   %4
                              59.0        86.0       -13.0       'Thermoelectric w/Forced Air'        '16-bit'              [ 2 1 ]
[ 1.1 0.364 0.180]          [ 0.9 2.7 5.6 ]          5.20     4.40     7.10      10.0        0       'N/A'       0           0     ;
%5
                              59.0        86.0       -76.0       'Thermoelectric w/Chilled Water'     '16-bit'      [ 4   2.5   2.1
1.6 ]    [ 6.5 1.90 1.18 0.55 ]   [ 0.15 0.52 0.85 1.8 ]      6.00     5.20     9.40      20.5        0       'N/A'       0
0     ;   %6
                              59.0        86.0       -76.0       'Thermoelectric w/Chilled Water'     '16-bit'      [ 4   2.5   2.1
1.6 ]    [ 6.5 1.90 1.18 0.55 ]   [ 0.15 0.52 0.85 1.8 ]      6.00     5.20     9.40      20.5        0       'N/A'       0
0     ;   %7
                                0           0         0.0                   'N/A'                       '12-bit'              [ 0 ]
[ 0 60 ]              [ 5 9 12 15 18 38 ]        5.30     5.30     2.30        0       0       'N/A'       0           0     ;   %8
                                0           0         0.0                   'N/A'                       '12-bit'              [ 0 ]
[ 0.0005 60 ]            [ 10 18 24 30 48 ]       5.30     5.30     2.50        0       0       'N/A'       0           0     ;
%9
                                0           0       -20.0       'Thermoelectric to -20 Ambient'      '12-bit'              [ 0 ]
[ 0 1200 ]            [ 5 9 12 15 18 38 ]        5.40     5.40     5.00        0       0       'N/A'       0           0     ;
%10
                             50.0       104.0           0     'Digital Temperature Compensation'   '12-bit'              [ 20 ]
[ 0.000005 65 ]             [ 12 23 ]              1.54     1.54     2.68      9.53        0       'N/A'       0           0     ;
%11
                             23.0       113.0           0                   'N/A'                       '10-bit'              [ 0.0157
]            [ 0.000001 0.5 ]               [ 0 ]              2.88     2.88     4.88      1.38        0       'N/A'       0
0     ;   %12
                           -148.0       104.0           0                   'N/A'                       'N/A'                 [ 2 1 ]
[ 1.1 0.364 0.180]          [ 0.9 2.7 5.6 ]          1.80     1.45    0.079        0       24        'DC'        0           0     ;
%13
                            -67.0       257.0           0                   'N/A'                       'N/A'                 [ 0 ]
[  0  ]                   [ 0 ]                 0         0         0         0       24        'DC'        5         0.17    ;   %14
                            -22.0       212.0           0                   'N/A'                       'N/A'                 [ 0 ]
[  0  ]                   [ 0 ]                2.88      2.08     0.24       0       24        'DC'        0           0     ;   %15
                              0           0             0                   'N/A'                       'N/A'                 [ 0 ]
[  0  ]                   [ 0 ]                1.60      1.60     0.435      0       22        'DC'        0           0     ;   %16
                              0           0             0                   'N/A'                       'N/A'                 [ 0 ]
[  0  ]                   [ 0 ]                2.00      2.00     0          0       24        'DC'        0           0     ;   %17
                           -130.0       104.0           0                   'N/A'                       'N/A'                 [ 0 ]
[  0  ]                   [ 0 ]                2.80      2.80     0.20       0       20        'DC'        0           0     ;   %18
                            -13.0       131.0           0                   'N/A'                       'N/A'                 [ 10 ]
[  0  ]                   [ 0 ]                0.90      0.25     0.293      0       15        'DC'        0         20.83   ;   %19
                            -13.0       158.0           0                   'N/A'                       'N/A'                 [ 20 ]
[  0  ]                   [ 0 ]                1.20      0.60     0.34       0       15        'DC'        0         0.581   };
%20

    ImgUnitLabelStr2 = { 'deg F'  'deg F'  'deg F'  'N/A'  'N/A'  'MHz'  'sec'  'fps'  'in'  'in'  'in'  'lbm'  'V'   'Type'
'mA'   'BTUs/hr'   };

end


% Comments on Features / Use
IMAGINGSENSOR_DB3{1,1} = {  'Ultra-Sensitive'   ;
                            'Fiber-optic coupled Camera'   ;
                            'Scientific Grade'   ;
```

```
                                'Low-noise, dual-speed, four-port readout architecture'   ;
                                'USB 2.0 Computer Interface'   };
IMAGINGSENSOR_DB3{2,1} = {      'Largest Commercially Available CCD'   ;
                                'Scientific X-ray imaging and electron imaging'   ;
                                'Ultra-Sensitive'   ;
                                'Fiber-optic coupled Camera'   ;
                                'Scientific Grade'   ;
                                'Low-noise, dual-speed, four-port readout architecture'   ;
                                'USB 2.0 Computer Interface'   };
IMAGINGSENSOR_DB3{3,1} = {      'Flight Rated, used on I-SPOT Mission'   ;
                                'Largest Commercially Available CCD'   ;
                                'Largest Commercially Available Taper 200 mm'   ;
                                'Scientific X-ray imaging and electron imaging'   ;
                                'Ultra-Sensitive'   ;
                                'Fiber-optic coupled Camera'   ;
                                'Scientific Grade'   ;
                                'Low-noise, dual-speed, four-port readout architecture'   ;
                                'USB 2.0 Computer Interface'
                                'Anti-blooming and integration control'   };
IMAGINGSENSOR_DB3{4,1} = {      'Multiport Readout'   ;
                                'Software Selectable Gain'   ;
                                'TTL Trigger I/O'   ;
                                'USB 2.0 interface'   ;
                                'High Optical Throughput'   ;
                                'Supports standard Fmount lenses and mounting hardware'   ;
                                'Low noise, multi-speed, multiport readout architecture'   };
IMAGINGSENSOR_DB3{5,1} = {      'Back-illuminated CCD option'   ;
                                'Highest quantum efficiency'   ;
                                'Four-port readout'   ;
                                'Optimal design for speed and sensitivity'   ;
                                'Dual Digitizer'   ;
                                'Separate, optimized readout channels for lowest noise and highest speed'   ;
                                'Deep Cooling & Low Noise Readout'   ;
                                'Software Gain & Binning'   ;
                                'Scientific precision and accuracy'    };
IMAGINGSENSOR_DB3{6,1} = {      'Back-illuminated CCD option'   ;
                                'Highest quantum efficiency'   ;
                                'Four-port readout'   ;
                                'Optimal design for speed and sensitivity'   ;
                                'Deep thermoelectric cooling'   ;
                                'Minimize dark noise'   ;
                                'High-performance low-noise electronics'   ;
                                'Minimize readout noise'   ;
                                'Scientific precision and accuracy'   ;
                                'Software-controlled binning & windowing'   ;
                                'Optimize speed versus resolution'   };
IMAGINGSENSOR_DB3{7,1} = {      'Back-illuminated CCD option'   ;
                                'Highest quantum efficiency'   ;
                                'Four-port readout'   ;
                                'Optimal design for speed and sensitivity'   ;
                                'Deep thermoelectric cooling'   ;
                                'Minimize dark noise'   ;
                                'High-performance low-noise electronics'   ;
                                'Minimize readout noise'   ;
```

```
                                'Scientific precision and accuracy'   ;
                                'Software-controlled binning & windowing'   ;
                                'Optimize speed versus resolution'   };
IMAGINGSENSOR_DB3{8,1} = {      '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'  ;
                                'ISO 9001:2000 CERTIFIED'  };
IMAGINGSENSOR_DB3{9,1} = {      'Microscope Monochrome Camera'   ;
                                '2-Megapixel Monochrome Imager'   ;
                                'Progressive Scan Interline CCD Array'   ;
                                'Optional Automated RGB 3-Pass Color Filter'   ;
                                'Software image editing tools'   ;
                                'IEEE 1394 FireWire 6-pin Computer Interface'   ;
                                'C-Mount Optical Interface'   ;
                                'Electronic Shutter: Integration time 500 micro-sec to 60 seconds'   ;
                                'ISO 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'   ;
                                'ISO 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. 8 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'   ;
```

```
                                'Scientific precision and accuracy'    };
IMAGINGSENSOR_DB3{14,1} = { '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'   ;
                            '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'    };
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);
```

```
ImgUnitLabelStr = cat(2,ImgUnitLabelStr1,ImgUnitLabelStr2);

IMG_MATRIX = IMAGINGSENSOR_DB;

%  Select Sensors By System Type - 'CAMERA' or 'ARRAY' or 'LINEAR'
tmp = 1;
 for typ = 1:size(IMG_MATRIX,1)
     if strcmpi(IMG_MATRIX{typ,2},SYS_TYPE) == 1
         for s = 1:size(IMG_MATRIX,2)
             IMG_MATRIX2{tmp,s} = IMG_MATRIX{typ,s};
         end
         tmp = tmp + 1;
     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('\n\n   ***  ERROR:  NULL DATABASE  ***');
                 fprintf('\n  No Sensor Exist in Database which Meet Imaging Requirements!');
                 fprintf('\n  Image Type:    %s', STR1);
                 IMG_MATRIX_TMP = IMG_MATRIX2;
             end
         end

         clear IMG_MATRIX2;
         IMG_MATRIX2 = IMG_MATRIX_TMP;
         clear IMG_MATRIX_TMP;
     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
         end
         if FLAG == 1
            c = c + 1;
        end
        FLAG = 0;
     end

     if exist('IMG_MATRIX_TMP') == 0
         if strcmpi(Print,'Y') == 1
             fprintf('\n\n   ***  ERROR:  NULL DATABASE  ***');
             fprintf('\n  No Sensor Exist in Database which Meet Imaging Requirements!');
             fprintf('\n  Image Type:    %s', STR1);
             IMG_MATRIX_TMP = IMG_MATRIX2;
         end
     end

     clear IMG_MATRIX2;
     IMG_MATRIX2 = IMG_MATRIX_TMP;
     clear 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
         end
         if FLAG == 1
            c = c + 1;
        end
        FLAG = 0;
     end

     if exist('IMG_MATRIX_TMP') == 0
         if strcmpi(Print,'Y') == 1
             fprintf('\n\n   ***  ERROR:  NULL DATABASE  ***');
             fprintf('\n  No Sensor Exist in Database which Meet Imaging Requirements!');
             fprintf('\n  Image Type:    %s', STR1);
```

```
                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};
                    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('\n\n   ***  ERROR:  NULL DATABASE  ***');
                fprintf('\n  No Sensor Exist in Database which Meet Imaging Requirements!');
                fprintf('\n  Image Type:    %s', STR1);
                IMG_MATRIX_TMP = IMG_MATRIX2;
            end
        end

        clear IMG_MATRIX2;
        IMG_MATRIX2 = IMG_MATRIX_TMP;
        clear IMG_MATRIX_TMP;
    end

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;
```

```
    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('\\n  ***  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
        end

    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


end

%  Copies IMG_MATRIX2 to IMG_MATRIX5 if all Sensors Eliminated
if exist('IMG_MATRIX5') == 0
    IMG_MATRIX5 = IMG_MATRIX2;
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('\n\n   ***   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   ***');
    fprintf('\n\n***   I N P U T S   ***');
```

```
fprintf('\n  System Type:           %s', SYS_TYPE);
fprintf('\n  Resolution:            %s', RESOLUTION);
fprintf('\n  Image Type:          ' );
for r = 1:length(IMG_TYPE)
    fprintf('   %s', IMG_TYPE{r} );
end
fprintf('\n  Unit System:           %s', UNITS);

fprintf('\n\n*********************************************************************************\n');

%  Imaging Sensor Properties
fprintf('\n\n** Sensor Properties **');
fprintf('\n  Imaging Sensor Type:              %s', IMG_MATRIX5{1,1});
fprintf('\n  Imaging Spectrum:              ' );
for r = 1:length(IMG_MATRIX4{1,3})
    fprintf('    %s', IMG_MATRIX4{1,3}{r} );
end
fprintf('\n  Array Dimensions - Length:      %10d   %s', IMG_MATRIX5{1,4}, ImgUnitLabelStr{4} );
fprintf('\n                       Width:      %10d   %s', IMG_MATRIX5{1,5}, ImgUnitLabelStr{5} );
fprintf('\n  Sensor Pixel Size:              %10.4f   %s', IMG_MATRIX5{1,6}, ImgUnitLabelStr{6} );
fprintf('\n  Imaging Array Size - Length:    %10.3f   %s', IMG_MATRIX5{1,7}, ImgUnitLabelStr{7} );
fprintf('\n                        Width:    %10.3f   %s', IMG_MATRIX5{1,8}, ImgUnitLabelStr{8} );
fprintf('\n  Read Noise at 1 MHz:            %10d   %s', IMG_MATRIX5{1,9}, ImgUnitLabelStr{9} );
fprintf('\n              250 kHz:            %10d   %s', IMG_MATRIX5{1,10}, ImgUnitLabelStr{10} );
fprintf('\n  Full Well Capacity - Pixel:     %10d   %s', IMG_MATRIX5{1,11}, ImgUnitLabelStr{11} );
fprintf('\n                     Register:    %10d   %s', IMG_MATRIX5{1,12}, ImgUnitLabelStr{12} );
fprintf('\n  Sensor Gain:                    %10.2f   %s', IMG_MATRIX5{1,13}, ImgUnitLabelStr{13} );
fprintf('\n  Sensor Linearity:               %10.2f   %s', IMG_MATRIX5{1,14}, ImgUnitLabelStr{14} );
fprintf('\n  ADC Dynamic Range:                  %s   %s', IMG_MATRIX5{1,19}, ImgUnitLabelStr{19} );
fprintf('\n  Readout Rates:             ');
for i = 1:length(IMG_MATRIX5{1,20})
    fprintf('    %7.4f', IMG_MATRIX5{1,20}(i) );
end
fprintf('    %s', ImgUnitLabelStr{20} );
fprintf('\n  Readout Time:              ');
for i = 1:length(IMG_MATRIX5{1,21})
    fprintf('    %9.6f', IMG_MATRIX5{1,21}(i) );
end
fprintf('    %s', ImgUnitLabelStr{21} );
fprintf('\n  Frame Rates:               ');
for i = 1:length(IMG_MATRIX5{1,22})
    fprintf('    %6.2f', IMG_MATRIX5{1,22}(i) );
end
fprintf('    %s', ImgUnitLabelStr{22} );

%  Environmental Properties
fprintf('\n\n   Environmental Properties');
fprintf('\n  Operating Temperature Range - Low:  %10.2f   %s', IMG_MATRIX5{1,15}, ImgUnitLabelStr{15} );
fprintf('\n                              High:  %10.2f   %s', IMG_MATRIX5{1,16}, ImgUnitLabelStr{16} );
fprintf('\n  Cooled Temperature:                %10.2f   %s', IMG_MATRIX5{1,17}, ImgUnitLabelStr{17} );
fprintf('\n  Cooling Method:                       %s   %s', IMG_MATRIX5{1,18}, ImgUnitLabelStr{18} );

fprintf('\n\n  Physical Properties');
fprintf('\n  Dimensions - Length:  %10.2f   %s', IMG_MATRIX5{1,23}, ImgUnitLabelStr{23} );
```

```
    fprintf('\n               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\n  Power Requirements');
    fprintf('\n  Input Voltage:                %9.3f   %s', IMG_MATRIX5{1,27}, ImgUnitLabelStr{27} );
    fprintf('\n  Input Voltage Type:               %s   %s', IMG_MATRIX5{1,28}, ImgUnitLabelStr{28} );
    fprintf('\n  Input Amperage:               %9.3f   %s', IMG_MATRIX5{1,29}, ImgUnitLabelStr{29} );
    fprintf('\n  Power Requirement:            %9.3f   %s', IMG_MATRIX5{1,30}, ImgUnitLabelStr{30} );

    fprintf('\n\n  Sensor Comments:');
    for r = 1:length(IMG_MATRIX5{1,31})
        fprintf('\n    %s', IMG_MATRIX5{1,31}{r} );
    end

end


%**************************************************************************
%***     Writes a text file summary of the Planetary Properties

    Fout = fopen('IMAGING_SENSOR_Summary.txt','w+');

    fprintf(Fout, '\n\n   ***   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   ***');
    fprintf(Fout, '\n\n***   I N P U T S   ***');
    fprintf(Fout, '\n  System Type:           %s', SYS_TYPE);
    fprintf(Fout, '\n  Resolution:            %s', RESOLUTION);
    fprintf(Fout, '\n  Image Type:          ' );
    for r = 1:length(IMG_TYPE)
        fprintf(Fout, '    %s', IMG_TYPE{r} );
    end
    fprintf(Fout, '\n  Unit System:           %s', UNITS);

    fprintf(Fout, '\n\n**************************************************************************\n');

    %  Imaging Sensor Properties
    fprintf(Fout, '\n\n** Sensor Properties **');
    fprintf(Fout, '\n  Imaging Sensor Type:              %s', IMG_MATRIX5{1,1});
    fprintf(Fout, '\n  Imaging Spectrum:               ' );
    for r = 1:length(IMG_MATRIX5{1,3})
        fprintf(Fout, '    %s', IMG_MATRIX5{1,3}{r} );
    end
    fprintf(Fout, '\n  Array Dimensions - Length:      %10d   %s', IMG_MATRIX5{1,4}, ImgUnitLabelStr{4} );
    fprintf(Fout, '\n                     Width:       %10d   %s', IMG_MATRIX5{1,5}, ImgUnitLabelStr{5} );
    fprintf(Fout, '\n  Sensor Pixel Size:              %10.4f   %s', IMG_MATRIX5{1,6}, ImgUnitLabelStr{6} );
    fprintf(Fout, '\n  Imaging Array Size - Length:    %10.3f   %s', IMG_MATRIX5{1,7}, ImgUnitLabelStr{7} );
    fprintf(Fout, '\n                       Width:    %10.3f   %s', IMG_MATRIX5{1,8}, ImgUnitLabelStr{8} );
    fprintf(Fout, '\n  Read Noise at 1 MHz:            %10d   %s', IMG_MATRIX5{1,9}, ImgUnitLabelStr{9} );
    fprintf(Fout, '\n               250 kHz:           %10d   %s', IMG_MATRIX5{1,10}, ImgUnitLabelStr{10} );
    fprintf(Fout, '\n  Full Well Capacity - Pixel:     %10d   %s', IMG_MATRIX5{1,11}, ImgUnitLabelStr{11} );
    fprintf(Fout, '\n                     Register:    %10d   %s', IMG_MATRIX5{1,12}, ImgUnitLabelStr{12} );
    fprintf(Fout, '\n  Sensor Gain:                    %10.2f   %s', IMG_MATRIX5{1,13}, ImgUnitLabelStr{13} );
    fprintf(Fout, '\n  Sensor Linearity:               %10.2f   %s', IMG_MATRIX5{1,14}, ImgUnitLabelStr{14} );
    fprintf(Fout, '\n  ADC Dynamic Range:                  %s   %s', IMG_MATRIX5{1,19}, ImgUnitLabelStr{19} );
```

```
fprintf(Fout, '\n  Readout Rates:            ');
for i = 1:length(IMG_MATRIX5{1,20})
    fprintf(Fout, '    %7.4f', IMG_MATRIX5{1,20}(i) );
end
fprintf(Fout, '    %s', ImgUnitLabelStr{20} );
fprintf(Fout, '\n  Readout Time:             ');
for i = 1:length(IMG_MATRIX5{1,21})
    fprintf(Fout, '    %9.6f', IMG_MATRIX5{1,21}(i) );
end
fprintf(Fout, '    %s', ImgUnitLabelStr{21} );
fprintf(Fout, '\n  Frame Rates:              ');
for i = 1:length(IMG_MATRIX5{1,22})
    fprintf(Fout, '    %6.2f', IMG_MATRIX5{1,22}(i) );
end
fprintf(Fout, '    %s', ImgUnitLabelStr{22} );


%  Environmental Properties
fprintf(Fout, '\n\n   Environmental Properties');
fprintf(Fout, '\n  Operating Temperature Range - Low:  %10.2f   %s', IMG_MATRIX5{1,15}, ImgUnitLabelStr{15} );
fprintf(Fout, '\n                               High:  %10.2f   %s', IMG_MATRIX5{1,16}, ImgUnitLabelStr{16} );
fprintf(Fout, '\n  Cooled Temperature:                 %10.2f   %s', IMG_MATRIX5{1,17}, ImgUnitLabelStr{17} );
fprintf(Fout, '\n  Cooling Method:                      %s   %s', IMG_MATRIX5{1,18}, ImgUnitLabelStr{18} );


fprintf(Fout, '\n\n  Physical Properties');
fprintf(Fout, '\n  Dimensions - Length:  %10.2f   %s', IMG_MATRIX5{1,23}, ImgUnitLabelStr{23} );
fprintf(Fout, '\n                Width:  %10.2f   %s', IMG_MATRIX5{1,24}, ImgUnitLabelStr{24} );
fprintf(Fout, '\n               Height:  %10.2f   %s', IMG_MATRIX5{1,25}, ImgUnitLabelStr{25} );
fprintf(Fout, '\n  Sensor Mass:          %10.3f   %s', IMG_MATRIX5{1,26}, ImgUnitLabelStr{26} );

fprintf(Fout, '\n\n  Power Requirements');
fprintf(Fout, '\n  Input Voltage:                %9.3f   %s', IMG_MATRIX5{1,27}, ImgUnitLabelStr{27} );
fprintf(Fout, '\n  Input Voltage Type:                %s   %s', IMG_MATRIX5{1,28}, ImgUnitLabelStr{28} );
fprintf(Fout, '\n  Input Amperage:               %9.3f   %s', IMG_MATRIX5{1,29}, ImgUnitLabelStr{29} );
fprintf(Fout, '\n  Power Requirement:            %9.3f   %s', IMG_MATRIX5{1,30}, ImgUnitLabelStr{30} );

fprintf(Fout, '\n\n  Sensor Comments:');
for r = 1:length(IMG_MATRIX5{1,31})
    fprintf(Fout, '\n    %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/ElectronicInclinometers.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  INCLINOMETERSENSOR_DB(IS_Type, Property)

%  Properties             [    SENSOR      INCLINATION RANGE        OUTPUT DATA      RESOLUTION   SENSITIVITY     NON-LINEARITY
VOLTAGE TYPE    VOLTAGE    RESPONSE TIME    BANDWIDTH       ]
```

```
%    Units              [      TYPE       Low    Deg    High          N/A              deg          mV/deg            %
N/A          Volts      sec                Hz              ]
INCLINOMETERSENSOR_DB1 =  {   'H4A1-30'    -30.00       30.00      'Analog 0-5V'        0.02         67.70             0.50
'DC'       [ 8   30 ]    0.30               3.00          ;
                           'H4A1-45'    -45.00       45.00      'Analog 0-5V'        0.03         44.40             0.50
'DC'       [ 8   30 ]    0.30               3.00          ;
                           'H4A1-70'    -70.00       70.00      'Analog 0-5V'        0.04         28.60             0.20
'DC'       [ 8   30 ]    0.30               3.00          ;
                           'H5A1-30'    -30.00       30.00   'Analog 0-5V, 0-10V'    0.02         67.70             0.50
'DC'       [ 12 30 ]     0.30               3.00          ;
                           'H5A1-45'    -45.00       45.00   'Analog 0-5V, 0-10V'    0.03         44.40             0.50
'DC'       [ 12 30 ]     0.30               3.00          ;
                           'H5A1-60'    -60.00       60.00   'Analog 0-5V, 0-10V'    0.03         33.30             0.20
'DC'       [ 12 30 ]     0.30               3.00          ;
                           'H5A1-90'    -90.00       90.00   'Analog 0-5V, 0-10V'    0.05         22.20             0.20
'DC'       [ 12 30 ]     0.30               3.00          ;
                             'N2'       -10.00       10.00       'Analog mV'         0.002        12.00             0.50
'DC'       [ 3    6 ]    0.30               3.00          ;
                             'N3'       -30.00       30.00       'Analog mV'         0.005         5.75             0.20
'DC'       [ 3    6 ]    0.30               3.00          ;
                             'N4'       -70.00       70.00       'Analog mV'         0.010         3.65             0.20
'DC'       [ 3    6 ]    0.30               3.00          ;
                            'NG2U'      -10.00       10.00      'Analog 0-5V'        0.001       200.00             0.10
'DC'       [ 8   30 ]    0.30               3.00          ;
                            'NG3U'      -30.00       30.00      'Analog 0-5V'        0.003        67.00             0.10
'DC'       [ 8   30 ]    0.30               3.00          ;
                            'NG4U'      -80.00       80.00      'Analog 0-5V'        0.010        25.00             0.10
'DC'       [ 8   30 ]    0.30               3.00          ;
                          'LCF 196'     -90.00       90.00      'Analog 0-5V'      0.00017          0               0.10
'DC'       [ 12 18 ]      0               30.00          ;
                          'LSRP-90'     -90.00       90.00      'Analog 0-5V'    0.0000573          0               0.05
'DC'       [ 12 18 ]      0               30.00          ;
                           'L-210'      -80.00       80.00       'Analog mV'         0.0083       100.00             1.25
'AC'       [    0   ]     0                 0            ;
                           'L-211U'     -60.00       60.00       'Analog mV'         0.0083       100.00             3.33
'AC'       [    0   ]     0                 0            ;
                           'L-212T'     -45.00       45.00       'Analog mV'         0.0056       150.00             2.22
'AC'       [    0   ]     0                 0            };


IncUnitLabelStr1 = { 'N/A'  'Deg'  'Deg'  'N/A'  'Deg'  'mV/deg'  '%'  'N/A'  'V'  'sec'  'Hz' };

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%   INCLINOMETER Sensor Data Base Created Based on un Unit System
%%  2 - D Matrix  INCLINOMETERSENSOR_DB(IS_Type, Property)

if strcmpi(UNITS,'SI') == 1
 %  Properties          [        OPERATING TEMP         STORAGE TEMP            DIMENSIONS  (mm)       WEIGHT   ]
 %    Units             [       Low  deg C  High      Low  deg C  High        Length   Width   Height    gram    ]
    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    ;
```

```
                                -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        48.8    48.8    21.6       110     ;
                                -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       80.0       -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


if strcmpi(UNITS,'British') == 1
 %  Properties          [         OPERATING TEMP          STORAGE TEMP              DIMENSIONS  (in)        WEIGHT    ]
 %    Units             {        Low  deg F  High       Low  deg F  High         Length   Width   Height       oz      ]
    INCLINOMETERSENSOR_DB2 = {   -40.0       185.0      -49.0       194.0         2.60    2.60     1.10       8.00    ;
                                 -40.0       185.0      -49.0       194.0         2.60    2.60     1.10       8.00    ;
                                 -40.0       185.0      -49.0       194.0         2.60    2.60     1.10       8.00    ;
                                 -40.0       185.0      -49.0       194.0         2.52    2.52     1.10       7.50    ;
                                 -40.0       185.0      -49.0       194.0         2.52    2.52     1.10       7.50    ;
                                 -40.0       185.0      -49.0       194.0         2.52    2.52     1.10       7.50    ;
                                 -40.0       185.0      -49.0       194.0         2.52    2.52     1.10       7.50    ;
                                 -40.0       185.0      -49.0       194.0         1.46    1.46     0.46       0.653   ;
                                 -40.0       185.0      -49.0       194.0         1.46    1.46     0.46       0.653   ;
                                 -40.0       185.0      -49.0       194.0         1.46    1.46     0.46       0.653   ;
                                 -40.0       185.0      -49.0       194.0         1.92    1.92     0.85       3.88    ;
                                 -40.0       185.0      -49.0       194.0         1.92    1.92     0.85       3.88    ;
                                 -40.0       185.0      -49.0       194.0         1.92    1.92     0.85       3.88    ;
                                 -40.0       176.0      -76.0       194.0         9.42    0.875    0.875     11.00    ;
                                  -0.4       159.8      -76.0       194.0         1.60    1.43     1.43       4.00    ;
                                 -65.2       255.2      -65.2       255.2         1.63    0.73     0.63       0       ;
                                 -65.2       255.2      -65.2       255.2         1.63    0.73     0.63       0       ;
                                 -65.2       255.2      -65.2       255.2         1.63    0.73     0.63       0       };

    IncUnitLabelStr2 = { 'deg F'  'deg F'  'deg F'  'deg F'  'in'  'in'  'in'  'oz'  };
end


% Comments on Features / Use
INCLINOMETERSENSOR_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'   ;
```

```
                                'EMC protected'    ;
                                'Mechanical Zero adjustment'  };
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'  };
INCLINOMETERSENSOR_DB3{3,1} = {  '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'  };
INCLINOMETERSENSOR_DB3{4,1} = {  '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'  };
INCLINOMETERSENSOR_DB3{5,1} = {  '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'  };
INCLINOMETERSENSOR_DB3{6,1} = {  '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'  };
INCLINOMETERSENSOR_DB3{7,1} = {  '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'  };
INCLINOMETERSENSOR_DB3{8,1} = {   '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{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'   ;
                                 'Seals meet MIL-STD 202, Method 112 Requirements'   };
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 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'   };
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 dampening 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 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'   };


%  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;
for I = 1:size(INCLINOMETERSENSOR_DB,1)
    if (INCLINOMETERSENSOR_DB{I,3} <= INC_ANGLE) && (INCLINOMETERSENSOR_DB{I,3} >= TOL_LOWER)
         for s = 1:size(INCLINOMETERSENSOR_DB,2)
             INCL_MATRIX{c,s} = INCLINOMETERSENSOR_DB{I,s};
         end
         c = c +1;
     end
 end


%  Continue Selection Process From Viable Options - If Multiple Sensor Options Exist
if size(INCL_MATRIX,1) > 1

    %  Select Via Temperature Operational Range
       c = 1;
       %  Build Up Database of Possible Inclinometer Sensors Based on Temperature Range
       for t = 1:size(INCL_MATRIX,1)
        if (PLTEMP >= INCL_MATRIX{t,12}) && (PLTEMP <= INCL_MATRIX{t,13})
            for a = 1:size(INCL_MATRIX,2)
                INCL_MATRIX2{c,a} = INCL_MATRIX{t,a};
            end
            c = c+1;
        end
       end

    %  Copies INCL_MATRIX to INCL_MATRIX2 if all Sensors Eliminated based on Temperature Range
    if exist('INCL_MATRIX2') == 0
        INCL_MATRIX2 = INCL_MATRIX;
    end

    if size(INCL_MATRIX2,1) >1

        %  Down Select by Max Temperature Range
              for d = 1:size(INCL_MATRIX2,1)
            TEMP_INCL_DELTA(d) = INCL_MATRIX2{d,13} - INCL_MATRIX2{d,12};
               end

               %   Find the Maximum Inclinometer Temperature Range Sensor
               MAX_TEMP_INCL__DELTA = max(TEMP_INCL_DELTA);

               %  For Multiple Sensors with the Same Range Reduce Sensor Matrix to Those
               %  with the Greatest Range
               k = 1;
               for c = 1:length(TEMP_INCL_DELTA)
            if TEMP_INCL_DELTA(c) == MAX_TEMP_INCL__DELTA
                for j = 1:size(INCL_MATRIX2,2)
                    INCL_MATRIX3{k,j} = INCL_MATRIX2{c,j};
                end
                k = k + 1;
            end
               end
```

```
    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
        k = 1;
        %  Search for Flight Rated Components
        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
```

```
    %  Copies INCL_MATRIX4 to INCL_MATRIX5 if all Sensors Eliminated based on Temperature Range
    if exist('INCL_MATRIX5') == 0
        INCL_MATRIX5 = INCL_MATRIX4;
    end

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' = 'Y'

if strcmpi(Print,'Y') == 1

    fprintf('\n\n   ***   I N C L I N O M E T E R   S E N S O R   S U M M A R Y   R E S U L T S   ***');
    fprintf('\n\n***   I N P U T S   ***');
    fprintf('\n  Inclination Angle Range:     %s', INCR{1});
    fprintf('\n  Unit System:           %s', UNITS);

    fprintf('\n\n*************************************************************************\n');

    %  Inclinometer Sensor Properties
    fprintf('\n\n** Sensor Properties **');
    fprintf('\n  Inclinometer Sensor Type:                %s', INCL_MATRIX5{1,1});
    fprintf('\n  Sensing Inclination Range - Low:    %10.3f   %s', INCL_MATRIX5{1,2}, IncUnitLabelStr{2} );
    fprintf('\n                             High:    %10.3f   %s', INCL_MATRIX5{1,3}, IncUnitLabelStr{3} );
    fprintf('\n  Output Data Format:                      %s', INCL_MATRIX5{1,4});
    fprintf('\n  Sensor Resolution:                  %10.7f   %s', INCL_MATRIX5{1,5}, IncUnitLabelStr{5} );
    fprintf('\n  Sensor Sensitivity:                 %10.3f   %s', INCL_MATRIX5{1,6}, IncUnitLabelStr{6} );
    fprintf('\n  Sensor Non-Linearity:               %10.4f   %s', INCL_MATRIX5{1,7}, IncUnitLabelStr{7} );
    fprintf('\n  Response Time:                      %10.3f   %s', INCL_MATRIX5{1,10}, IncUnitLabelStr{10} );
    fprintf('\n  Sensor Bandwidth:                   %10.2f   %s', INCL_MATRIX5{1,11}, IncUnitLabelStr{11} );


    %  Environmental Properties
    fprintf('\n\n   Environmental Properties');
    fprintf('\n  Operating Temperature Range - Low:  %9.3f   %s', INCL_MATRIX5{1,12}, IncUnitLabelStr{12} );
    fprintf('\n                                High:  %9.3f   %s', INCL_MATRIX5{1,13}, IncUnitLabelStr{13} );
    fprintf('\n  Storage Temperature Range - Low:    %9.3f   %s', INCL_MATRIX5{1,14}, IncUnitLabelStr{14} );
    fprintf('\n                             High:    %9.3f   %s', INCL_MATRIX5{1,15}, IncUnitLabelStr{15} );

    fprintf('\n\n  Power Requirements');
    fprintf('\n  Input Voltage Type:                          %s   %s', INCL_MATRIX5{1,8}, IncUnitLabelStr{8} );
    fprintf('\n  Input Voltage:                 ');
    for i = 1:length(INCL_MATRIX5{1,9})
        fprintf('    %2d', INCL_MATRIX5{1,9}(i) );
    end
    fprintf('    %s', IncUnitLabelStr{9} );
```

```
    fprintf('\n\n  Physical Properties');
    fprintf('\n  Dimensions - Length:  %10.3f   %s', INCL_MATRIX5{1,16}, IncUnitLabelStr{16} );
    fprintf('\n                Width:  %10.3f   %s', INCL_MATRIX5{1,17}, IncUnitLabelStr{17} );
    fprintf('\n               Height:  %10.3f   %s', INCL_MATRIX5{1,18}, IncUnitLabelStr{18} );
    fprintf('\n  Sensor Mass:          %10.3f   %s', INCL_MATRIX5{1,19}, IncUnitLabelStr{19} );


    fprintf('\n\n  Sensor Comments:');
    for r = 1:length(INCL_MATRIX5{1,20})
        fprintf('\n    %s', INCL_MATRIX5{1,20}{r} );
    end

end


%**************************************************************************
%***     Writes a text file summary of the Planetary Properties

    Fout = fopen('INCLINOMETER_SENSOR_Summary.txt','w+');

    fprintf(Fout, '\n\n   ***   I N C L I N O M E T E R   S E N S O R   S U M M A R Y   R E S U L T S   ***');
    fprintf(Fout, '\n\n***   I N P U T S   ***');
    fprintf(Fout, '\n  Inclination Angle Range:     %s', INCR{1});
    fprintf(Fout, '\n  Unit System:           %s', UNITS);

    fprintf(Fout, '\n\n*************************************************************************\n');

    %  Inclinometer Sensor Properties
    fprintf(Fout, '\n\n** Sensor Properties **');
    fprintf(Fout, '\n  Inclinometer Sensor Type:                %s', INCL_MATRIX5{1,1});
    fprintf(Fout, '\n  Sensing Inclination Range - Low:    %10.3f   %s', INCL_MATRIX5{1,2}, IncUnitLabelStr{2} );
    fprintf(Fout, '\n                              High:    %10.3f   %s', INCL_MATRIX5{1,3}, IncUnitLabelStr{3} );
    fprintf(Fout, '\n  Output Data Format:                      %s', INCL_MATRIX5{1,4});
    fprintf(Fout, '\n  Sensor Resolution:                       %10.7f   %s', INCL_MATRIX5{1,5}, IncUnitLabelStr{5} );
    fprintf(Fout, '\n  Sensor Sensitivity:                      %10.3f   %s', INCL_MATRIX5{1,6}, IncUnitLabelStr{6} );
    fprintf(Fout, '\n  Sensor Non-Linearity:                    %10.4f   %s', INCL_MATRIX5{1,7}, IncUnitLabelStr{7} );
    fprintf(Fout, '\n  Response Time:                           %10.3f   %s', INCL_MATRIX5{1,10}, IncUnitLabelStr{10} );
    fprintf(Fout, '\n  Sensor Bandwidth:                        %10.2f   %s', INCL_MATRIX5{1,11}, IncUnitLabelStr{11} );


    %  Environmental Properties
    fprintf(Fout, '\n\n   Environmental Properties');
    fprintf(Fout, '\n  Operating Temperature Range - Low:  %9.3f   %s', INCL_MATRIX5{1,12}, IncUnitLabelStr{12} );
    fprintf(Fout, '\n                                High:  %9.3f   %s', INCL_MATRIX5{1,13}, IncUnitLabelStr{13} );
    fprintf(Fout, '\n  Storage Temperature Range - Low:    %9.3f   %s', INCL_MATRIX5{1,14}, IncUnitLabelStr{14} );
    fprintf(Fout, '\n                               High:    %9.3f   %s', INCL_MATRIX5{1,15}, IncUnitLabelStr{15} );

    fprintf(Fout, '\n\n  Power Requirements');
    fprintf(Fout, '\n  Input Voltage Type:                          %s   %s', INCL_MATRIX5{1,8}, IncUnitLabelStr{8} );
    fprintf(Fout, '\n  Input Voltage:                  ');
    for i = 1:length(INCL_MATRIX5{1,9})
        fprintf(Fout, '    %2d', INCL_MATRIX5{1,9}(i) );
    end
    fprintf(Fout, '    %s', IncUnitLabelStr{9} );
```

```
    fprintf(Fout, '\n\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\n  Sensor Comments:');
    for r = 1:length(INCL_MATRIX5{1,20})
        fprintf(Fout, '\n    %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/optec_041.htm

function NEPHELOMETER_SENSORS(UNITS,Print)


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


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%   Nephelometer Sensor Data Base Created Based on un Unit System
```

```
%%  2 - D Matrix  NEPHELOMETER_DB(N_Type, Property)

if strcmpi(UNITS,'SI') == 1
 % Properties       [  Sensor       Wavelength1   Wavelength2  Wavelenght3  Bandwidth  Weight   Power Voltage  Width  Height
Length  Avg Time   Drift          Angle Integration      ]
 % Units            [    #              nm            nm           nm         kg       W       V       mm     mm
mm     sec    per m @ 60 sec    low deg   high deg      ]
    NEPHELOMETER_DB = {   'TSI-3563'      450          550          700        40       18      175      24      300    1100
250     4096     2.000E-07          7        170          ;
                         'NGN-2a'         320          550          730        100     12.247    62.5     13.8     202     419
271      600         0              5        175          ;
                         'NGN-3a'         0            550            0        200     12.247    62.5     13.8     223     419
271      810         0              5        175          };

    % Properties       [  Response Time  Rec Flow Rate     Operating Temperature     Ambient Rel Humidity ]
    % Units            [    sec          L/min     L/min    Low deg C  High deg C    Low %       High %   ]
    NEPHELOMETER_DB2 =  {    10           20        200       10          40          0           95      ;
                              4            2          2      -20          45          0            0      ;
                              4            7          9      -10          45          0          100      };

    NephUnitLabelStr1 = { 'sensor'  'nm'  'nm' 'nm' 'nm' 'kg' 'W' 'V' 'mm' '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      [    Weight   Power   Type  Voltage  Width  Height  Length     ]
       % Units           [     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     };

          NephPSLabelStr  = { 'kg' 'W' 'N/A'  'V' 'mm' 'mm' 'mm'  };

end

if strcmpi(UNITS,'BRITISH') == 1
 % {Properties      [    Sensor      Wavelength1  Wavelength2 Wavelenght3    Bandwith     Weight   Power     Voltage  Width
Height  Length  Avg Time   Drift          Angle Integration        ]
 % Units            [       #             inch         inch          inch          inch         lb     BTU's/hr     V      inch
inch    inch     sec    per ft @ 60 sec   low deg   high deg
    NEPHELOMETER_DB = {  'TSI-3563'   1.7716E-05   2.1653E-05   2.7559E-05    1.574E-06    39.68   597.656     24      11.81
9.84    43.3     4096    6.096E-08          7         170          ;
                         'NGN-2a'     1.2598E-05   2.1653E-05   2.8740E-05    3.937E-06     27     213.449     13.8    10.7
16.5    8.2      600           0             5         175          ;
                         'NGN-3a'          0       2.1653E-05        0        7.874E-06     27     213.449     13.8    10.7
16.5    8.2      810           0             5         175         };

    % Properties       [  Response Time  Rec Flow Rate     Operating Temperature     Ambient Rel Humidity ]
    % Units            [    sec          gal/min  gal/min    Low deg F  High deg F    Low %       High %   ]
    NEPHELOMETER_DB2 =  {    10          5.28344  52.8344      50          104          0           95      ;
                              4          0.528344 0.528344     -4          113          0            0      ;
                              4           1.8492   2.37755      14          113          0          100      };
```

```
    NephUnitLabelStr1 = { 'N/A' 'inch' 'inch' 'inch' 'inch' 'lb' 'BTUs/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     ]
       % Units           [     lb      BTU/hr   N/A    V        in     in      in       ]
            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
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) = 9.99E99;
        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('NEPHELOMETER_SENSOR_Data.mat');

save('NEPHELOMETER_SENSOR_Final.mat','N_MATRIX3','NephUnitLabelStr');


%*************************************************************************************************************************
%
%   ***   PRINT SUMMARY Output Section for 'Print' = 'Y'

if strcmpi(Print,'Y') == 1

    fprintf('\n\n   ***   N E P H E L O M E T E R   S E N S O R   S U M M A R Y   R E S U L T S   ***');
    fprintf('\n\n***   I N P U T S   ***');
    fprintf('\n  Unit System:           %s', UNITS);

    fprintf('\n\n*************************************************************************\n');

    %  Nephelometer Sensor Properties
    fprintf('\n\n** Sensor Properties **');
    fprintf('\n  Nephelometer Sensor Type:              %s', N_MATRIX3{1,1});
    fprintf('\n  Sensing Wavelengths - Min:    %12.4E   %s', N_MATRIX3{1,2}, NephUnitLabelStr{2} );
    fprintf('\n                      Middle:   %12.4E   %s', N_MATRIX3{1,3}, NephUnitLabelStr{3} );
    fprintf('\n                         Max:   %12.4E   %s', N_MATRIX3{1,4}, NephUnitLabelStr{4} );
    fprintf('\n  Sensing Bandwidth:             %12.4E   %s', N_MATRIX3{1,5}, NephUnitLabelStr{5} );
    fprintf('\n  Sampling Time:                 %12.2f   %s', N_MATRIX3{1,12}, NephUnitLabelStr{12} );
    fprintf('\n  Sensing Drift Rate:            %12.3E   %s', N_MATRIX3{1,13}, NephUnitLabelStr{13} );
    fprintf('\n  Integration Angles - Min:      %12.2f   %s', N_MATRIX3{1,14}, NephUnitLabelStr{14} );
    fprintf('\n                       Max:      %12.2f   %s', N_MATRIX3{1,15}, NephUnitLabelStr{15} );
    fprintf('\n  Response Time:                 %12.2f   %s', N_MATRIX3{1,16}, NephUnitLabelStr{16} );
    fprintf('\n  Sampling Flow Rate - Min:      %12.2f   %s', N_MATRIX3{1,17}, NephUnitLabelStr{17} );
    fprintf('\n                       Max:      %12.2f   %s', N_MATRIX3{1,18}, NephUnitLabelStr{18} );


    fprintf('\n\n  Power Requirements');
    fprintf('\n  Power Source:          %12.3f   %s', N_MATRIX3{1,7}, NephUnitLabelStr{7} );
    fprintf('\n  Input Voltage:         %12.3f   %s', N_MATRIX3{1,8}, NephUnitLabelStr{8} );


    fprintf('\n\n  Physical Properties');
    fprintf('\n  Sensor Mass:                   %12.2f   %s', N_MATRIX3{1,6}, NephUnitLabelStr{6} );
    fprintf('\n  Sensor Dimensions - Length:    %12.2f   %s', N_MATRIX3{1,11}, NephUnitLabelStr{11} );
    fprintf('\n                       Width:    %12.2f   %s', N_MATRIX3{1,9}, NephUnitLabelStr{9} );
    fprintf('\n                      Height:    %12.2f   %s', N_MATRIX3{1,10}, NephUnitLabelStr{10} );


    %  Environmental Properties
```

```
    fprintf('\n\n   Environmental Properties');
    fprintf('\n  Temperature Range - Low:      %12.2f   %s', N_MATRIX3{1,19}, NephUnitLabelStr{19} );
    fprintf('\n                     High:     %12.2f   %s', N_MATRIX3{1,20}, NephUnitLabelStr{20} );
    fprintf('\n  Relative Humidity - Min:      %12.2f   %s', N_MATRIX3{1,21}, NephUnitLabelStr{21} );
    fprintf('\n                      Max:      %12.2f   %s', N_MATRIX3{1,22}, NephUnitLabelStr{22} );


    fprintf('\n\n  Sensor Comments:');
    fprintf('\n    %s', N_MATRIX3{1,23} );


    %  Power Supply Properties
    fprintf('\n\n  Power Supply Properties');
    fprintf('\n  Power Generated:             %12.3f   %s', NephPS_DB{n,2}, NephPSLabelStr{2} );
    fprintf('\n  Supplied Voltage:            %12.2f   %s', NephPS_DB{n,4}, NephPSLabelStr{4} );
    fprintf('\n  Voltage Type:                        %s   %s', NephPS_DB{n,3}, NephPSLabelStr{3} );
    fprintf('\n  Sensor Mass:                 %12.2f   %s', NephPS_DB{n,1}, NephPSLabelStr{1} );
    fprintf('\n  Sensor Dimensions - Length:  %12.2f   %s', NephPS_DB{n,7}, NephPSLabelStr{7} );
    fprintf('\n                       Width:  %12.2f   %s', NephPS_DB{n,5}, NephPSLabelStr{5} );
    fprintf('\n                      Height:  %12.2f   %s', NephPS_DB{n,6}, NephPSLabelStr{6} );

end


%************************************************************************
%***     Writes a text file summary of the Planetary Properties

    Fout = fopen('NEPHELOMETER_SENSOR_Summary.txt','w+');

    fprintf(Fout, '\n\n   ***   N E P H E L O M E T E R   S E N S O R   S U M M A R Y   R E S U L T S   ***');
    fprintf(Fout, '\n\n***   I N P U T S   ***');
    fprintf(Fout, '\n  Unit System:           %s', UNITS);

    fprintf(Fout, '\n\n************************************************************************\n');

    %  Nephelometer Sensor Properties
    fprintf(Fout, '\n\n** Sensor Properties **');
    fprintf(Fout, '\n  Nephelometer Sensor Type:              %s', N_MATRIX3{1,1});
    fprintf(Fout, '\n  Sensing Wavelengths - Min:   %12.4E   %s', N_MATRIX3{1,2}, NephUnitLabelStr{2} );
    fprintf(Fout, '\n                      Middle:   %12.4E   %s', N_MATRIX3{1,3}, NephUnitLabelStr{3} );
    fprintf(Fout, '\n                         Max:   %12.4E   %s', N_MATRIX3{1,4}, NephUnitLabelStr{4} );
    fprintf(Fout, '\n  Sensing Bandwidth:           %12.4E   %s', N_MATRIX3{1,5}, NephUnitLabelStr{5} );
    fprintf(Fout, '\n  Sampling Time:               %12.2f   %s', N_MATRIX3{1,12}, NephUnitLabelStr{12} );
    fprintf(Fout, '\n  Sensing Drift Rate:          %12.3E   %s', N_MATRIX3{1,13}, NephUnitLabelStr{13} );
    fprintf(Fout, '\n  Integration Angles - Min:    %12.2f   %s', N_MATRIX3{1,14}, NephUnitLabelStr{14} );
    fprintf(Fout, '\n                       Max:    %12.2f   %s', N_MATRIX3{1,15}, NephUnitLabelStr{15} );
    fprintf(Fout, '\n  Response Time:               %12.2f   %s', N_MATRIX3{1,16}, NephUnitLabelStr{16} );
    fprintf(Fout, '\n  Sampling Flow Rate - Min:    %12.2f   %s', N_MATRIX3{1,17}, NephUnitLabelStr{17} );
    fprintf(Fout, '\n                       Max:    %12.2f   %s', N_MATRIX3{1,18}, NephUnitLabelStr{18} );


    fprintf(Fout, '\n\n  Power Requirements');
    fprintf(Fout, '\n  Power Source:          %12.3f   %s', N_MATRIX3{1,7}, NephUnitLabelStr{7} );
```

```
fprintf(Fout, '\n  Input Voltage:        %12.3f   %s', N_MATRIX3{1,8}, NephUnitLabelStr{8} );


fprintf(Fout, '\n\n  Physical Properties');
fprintf(Fout, '\n  Sensor Mass:                    %12.2f   %s', N_MATRIX3{1,6}, NephUnitLabelStr{6} );
fprintf(Fout, '\n  Sensor Dimensions - Length:     %12.2f   %s', N_MATRIX3{1,11}, NephUnitLabelStr{11} );
fprintf(Fout, '\n                       Width:     %12.2f   %s', N_MATRIX3{1,9}, NephUnitLabelStr{9} );
fprintf(Fout, '\n                      Height:     %12.2f   %s', N_MATRIX3{1,10}, NephUnitLabelStr{10} );


%  Environmental Properties
fprintf(Fout, '\n\n   Environmental Properties');
fprintf(Fout, '\n  Temperature Range - Low:      %12.2f   %s', N_MATRIX3{1,19}, NephUnitLabelStr{19} );
fprintf(Fout, '\n                     High:      %12.2f   %s', N_MATRIX3{1,20}, NephUnitLabelStr{20} );
fprintf(Fout, '\n  Relative Humidity - Min:      %12.2f   %s', N_MATRIX3{1,21}, NephUnitLabelStr{21} );
fprintf(Fout, '\n                      Max:      %12.2f   %s', N_MATRIX3{1,22}, NephUnitLabelStr{22} );


fprintf(Fout, '\n\n  Sensor Comments:');
fprintf(Fout, '\n    %s', N_MATRIX3{1,23} );


%  Power Supply Properties
fprintf(Fout, '\n\n  Power Supply Properties');
fprintf(Fout, '\n  Power Generated:             %12.3f   %s', NephPS_DB{n,2}, NephPSLabelStr{2} );
fprintf(Fout, '\n  Supplied Voltage:            %12.2f   %s', NephPS_DB{n,4}, NephPSLabelStr{4} );
fprintf(Fout, '\n  Voltage Type:                        %s   %s', NephPS_DB{n,3}, NephPSLabelStr{3} );
fprintf(Fout, '\n  Sensor Mass:                 %12.2f   %s', NephPS_DB{n,1}, NephPSLabelStr{1} );
fprintf(Fout, '\n  Sensor Dimensions - Length:  %12.2f   %s', NephPS_DB{n,7}, NephPSLabelStr{7} );
fprintf(Fout, '\n                       Width:  %12.2f   %s', NephPS_DB{n,5}, NephPSLabelStr{5} );
fprintf(Fout, '\n                      Height:  %12.2f   %s', NephPS_DB{n,6}, NephPSLabelStr{6} );

fclose(Fout);
```

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.omron.com/ecb/products/sensor/8.html
%    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
%    http://www.mksinst.com/product/catalog.aspx?catalogID=4
%
%

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') == 1
  %                                        PERFORMANCE CHARACTERISTICS @ 77
ENVIRONMANTAL DATA                                                    ]
 %  Properties      [       SENSOR     ACCURACY   STABILITY    OVER PRESSURE   BURST PRESSURE  LIFE CYCLE     PRESSURE RANGE
PROOF      BURST      SHOCK  IMPULSE   VIB SHOCK  FREQUENCY RANGE     HUMIDITY RANGE  ]
 %    Units                 TYPE         %        % @ 1 yr     @ Mx Rated       @ Mx Rated      Rated #       Low  kPa   High
kPa         kPa        g's    sec          g's     Low  Hz   High      Low  %RH  High  ]
    PRESSSENSOR_DB1 = { 'Ser 48-0025'    0.5        0.25           2.0             5.0          100E+06       0.0      172.369
344.738     861.845      100    0.011       20.0       20       2000       0.0     95.0    ;   %  1
                        'Ser 48-0050'    0.5        0.25           2.0             5.0          100E+06       0.0      344.738
689.476    1723.690      100    0.011       20.0       20       2000       0.0     95.0    ;   %  2
                        'Ser 48-0100'    0.5        0.25           2.0             5.0          100E+06       0.0      689.476
1378.950    3447.380      100    0.011       20.0       20       2000       0.0     95.0    ;   %  3
                        'Ser 48-0200'    0.5        0.25           2.0             5.0          100E+06       0.0     1378.950
2757.900    6894.760      100    0.011       20.0       20       2000       0.0     95.0    ;   %  4
                        'Ser 48-1000'    0.5        0.25           2.0             5.0          100E+06       0.0     6894.760
13789.500  27579.000      100    0.011       20.0       20       2000       0.0     95.0    ;   %  5
                       '19(C,U)-003P'    2.0        0.60           3.0             5.0            1E+06       0.0       20.684
62.053     103.421      100    0.011       10.0       20       2000       0.0      0.0    ;   %  6
                       '19(C,U)-005P'    2.0        0.60           3.0             5.0            1E+06       0.0       34.474
103.421     172.369      100    0.011       10.0       20       2000       0.0      0.0    ;   %  7
                       '19(C,U)-010P'    2.0        0.60           3.0             5.0            1E+06       0.0       68.948
206.843     344.738      100    0.011       10.0       20       2000       0.0      0.0    ;   %  8
```

```
                    '19(C,U)-015P'    2.0         0.60          3.0              5.0              1E+06      0.0      103.421
310.264    517.107      100    0.011       10.0      20      2000       0.0      0.0     ;   % 9
                    '19(C,U)-030P'    2.0         0.60          3.0              5.0              1E+06      0.0      206.843
620.528   1034.210      100    0.011       10.0      20      2000       0.0      0.0     ;   % 10
                    '19(C,U)-050P'    2.0         0.60          3.0              5.0              1E+06      0.0      344.738
1034.210   1723.690      100    0.011       10.0      20      2000       0.0      0.0     ;   % 11
                    '19(C,U)-100P'    2.0         0.60          3.0              5.0              1E+06      0.0      689.476
2068.430   3447.380      100    0.011       10.0      20      2000       0.0      0.0     ;   % 12
                    '19(C,U)-200P'    2.0         0.60          3.0              5.0              1E+06      0.0     1378.950
4136.850   6894.760      100    0.011       10.0      20      2000       0.0      0.0     ;   % 13
                    '19(C,U)-300P'    2.0         0.60          3.0              5.0              1E+06      0.0     2068.430
6205.280  10342.100      100    0.011       10.0      20      2000       0.0      0.0     ;   % 14
                    '19(C,U)-500P'    2.0         0.60          2.4              4.8              1E+06      0.0     3447.380
8273.710  16547.400      100    0.011       10.0      20      2000       0.0      0.0     ;   % 15
                    'OMRON E8Y'       3.0         0.0                            25.00              0          0.0        2.00
0.0       50.00       30     0.0        10.0      10      150      25.0     85.0     ;   % 16
                    'OMRON E8Y'       3.0         0.0                            10.00              0          0.0        5.00
0.0       50.00       30     0.0        10.0      10      150      25.0     85.0     ;   % 17
                    '13(C,U)0500P'    2.0         0.60          3.0              5.0              1E+06      0.0     3447.380
10342.10   17236.90      100    0.011       10.0      20      2000       0.0      0.0     ;   % 18
                    '13(C,U)1000P'    2.0         0.60          3.0              5.0              1E+06      0.0     6894.760
20684.30   34473.80      100    0.011       10.0      20      2000       0.0      0.0     ;   % 19
                    '13(C,U)2000P'    2.0         0.60          3.0              5.0              1E+06      0.0    13789.500
41368.50   68947.60      100    0.011       10.0      20      2000       0.0      0.0     ;   % 20
                    '13(C,U)3000P'    2.0         0.60          3.0              3.333            1E+06      0.0    20684.300
62052.80   68947.60      100    0.011       10.0      20      2000       0.0      0.0     ;   % 21
                    '13(C,U)5000P'    2.0         0.60          2.0              2.0              1E+06      0.0    34473.800
68947.60   68947.60      100    0.011       10.0      20      2000       0.0      0.0     ;   % 22
                    'PDCR 3500'       0.1         0.30          2.0              6.0              0          5.0     6894.76
13789.50   20684.30       0      0.0         0.0       0        0         0.0      0.0     ;   % 23
                    'PDCR 3500'       0.1         0.30          2.0              4.0              0          5.0    68947.60
68947.60   68947.60       0      0.0         0.0       0        0         0.0      0.0     ;   % 24
                    '2403SAT'       0.20          0.10          2.0              3.0              0          0.0    34473.79
68947.58  103421.37       30    0.011       25.0        0        0         0.0      0.0     ;   % 25

                  };

 %  Properties       [    OPERATING TEMP      STORAGE TEMP      VOLTAGE   CURRENT   BANDWIDTH          MATERIAL
DIMENSIONS  (mm)      WEIGHT       TEMPERATURE SENSOR PROPERTIES   ACCURACY    ]
 %    Units                Low deg C High    Low deg C High        V        mA          dB       SENSOR           CASE
Length  Width  Height     gram        Low   SENSING RANGE  High        deg C       ]
    PRESSSENSOR_DB2 = {  -40.0    104.44   -51.1    121.11     5.0       5.0          -3.0   'Stainless Steel' 'Stainless Steel'
48.56  22.23  22.23       0.0          -53.9                  121.1       1.1      ;   %  1
                        -40.0    104.44   -51.1    121.11     5.0       5.0          -3.0   'Stainless Steel' 'Stainless Steel'
48.56  22.23  22.23       0.0          -53.9                  121.1       1.1      ;   %  2
                        -40.0    104.44   -51.1    121.11     5.0       5.0          -3.0   'Stainless Steel' 'Stainless Steel'
48.56  22.23  22.23       0.0          -53.9                  121.1       1.1      ;   %  3
                        -40.0    104.44   -51.1    121.11     5.0       5.0          -3.0   'Stainless Steel' 'Stainless Steel'
48.56  22.23  22.23       0.0          -53.9                  121.1       1.1      ;   %  4
                        -40.0    104.44   -51.1    121.11     5.0       5.0          -3.0   'Stainless Steel' 'Stainless Steel'
48.56  22.23  22.23       0.0          -53.9                  121.1       1.1      ;   %  5
                        -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      ;   %  6
```

```
                          -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      ;   %  7
                          -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      ;   %  8
                          -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      ;   %  9
                          -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      ;   % 10
                          -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      ;   % 11
                          -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      ;   % 12
                          -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      ;   % 13
                          -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      ;   % 14
                          -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      ;   % 15
                          -10.0      55.00   -25.0      65.00    30.0     50.0            0   'Resin pipe/Zinc'         'PBT'
31.00  30.00  30.00      79.38            0                      0          0      ;   % 16
                          -10.0      55.00   -25.0      65.00    30.0     50.0            0   'Resin pipe/Zinc'         'PBT'
31.00  30.00  30.00      79.38            0                      0          0      ;   % 17
                          -40.0     125.00   -40.0     125.00    15.0      2.0            0   'Stainless Steel' 'Stainless Steel'
81.03  19.05  19.05       0.0             0                      0          0      ;   % 18
                          -40.0     125.00   -40.0     125.00    15.0      2.0            0   'Stainless Steel' 'Stainless Steel'
81.03  19.05  19.05       0.0             0                      0          0      ;   % 19
                          -40.0     125.00   -40.0     125.00    15.0      2.0            0   'Stainless Steel' 'Stainless Steel'
81.03  19.05  19.05       0.0             0                      0          0      ;   % 20
                          -40.0     125.00   -40.0     125.00    15.0      2.0            0   'Stainless Steel' 'Stainless Steel'
81.03  19.05  19.05       0.0             0                      0          0      ;   % 21
                          -40.0     125.00   -40.0     125.00    15.0      2.0            0   'Stainless Steel' 'Stainless Steel'
81.03  19.05  19.05       0.0             0                      0          0      ;   % 22
                          -53.9     121.11   -53.9     125.00     5.0      0.0            0   'Stainless Steel' 'Stainless Steel'
92.71  25.40  25.40       0.0             0                      0          0      ;   % 23
                          -53.9     121.11   -53.9     125.00    15.0      2.0            0   'Stainless Steel' 'Stainless Steel'
102.87  25.40  25.40       0.0             0                      0          0      ;   % 24
                          -54.0     121.00   -54.0     121.00    36.0      0.0            0   'Stainless Steel' 'Stainless Steel'
88.90  31.75  31.75     170.0             0                      0          0      ;   % 25
    };


    PressUnitLabelStr1 = {  'Type'   '%'   '% @ 1 yr'   'N/A'   'N/A'   'Cycles'   'kPa'  'kPa'  'kPa'   'kPa'   'g''s'   'sec'
'g''s'  'Hz'   'Hz' '% RH'   '% RH'   };
    PressUnitLabelStr2 = {  'deg C'  'deg C'  'deg C'  'deg C'  'V'   'mA'   'dB'   'N/A'   'N/A'   'mm'  'mm'   'mm'   'gr'
'deg C'  'deg C'   'deg C'    };

end

if strcmpi(UNITS,'British') == 1
 %                                    PERFORMANCE CHARACTERISTICS @ 77
ENVIRONMANTAL DATA                                                 ]
 %  Properties      [       SENSOR     ACCURACY   STABILITY    OVER PRESSURE   BURST PRESSURE  LIFE CYCLE    PRESSURE RANGE
PROOF    BURST      SHOCK  IMPULSE   VIB SHOCK  FREQUENCY RANGE     HUMIDITY RANGE  ]
```

```
 %    Units          [        TYPE          %        % @ 1 yr     @ Mx Rated        @ Mx Rated     Rated #       Low  psig  High
psig    psig       g's     sec            g's     Low  Hz   High     Low  %RH  High  ]
    PRESSSENSOR_DB1 = {  'Ser 48-0025'     0.5       0.25            2.0              5.0          100E+06       0.0         25.0
50.0    125.0      100     0.011         20.0       20        2000     0.0     95.0     ;   % 1
                         'Ser 48-0050'     0.5       0.25            2.0              5.0          100E+06       0.0         50.0
100.0    250.0      100     0.011         20.0       20        2000     0.0     95.0     ;   % 2
                         'Ser 48-0100'     0.5       0.25            2.0              5.0          100E+06       0.0        100.0
200.0    500.0      100     0.011         20.0       20        2000     0.0     95.0     ;   % 3
                         'Ser 48-0200'     0.5       0.25            2.0              5.0          100E+06       0.0        200.0
400.0   1000.0      100     0.011         20.0       20        2000     0.0     95.0     ;   % 4
                         'Ser 48-1000'     0.5       0.25            2.0              5.0          100E+06       0.0       1000.0
2000.0   4000.0      100     0.011         20.0       20        2000     0.0     95.0     ;   % 5
                         '19(C,U)-003P'    2.0       0.60            3.0              5.0            1E+06       0.0          3.0
9.0     15.0       100     0.011        10.0       20        2000     0.0      0.0    ;   % 6
                         '19(C,U)-005P'    2.0       0.60            3.0              5.0            1E+06       0.0          5.0
15.0     25.0       100     0.011         10.0       20        2000     0.0      0.0    ;   % 7
                         '19(C,U)-010P'    2.0       0.60            3.0              5.0            1E+06       0.0         10.0
30.0     50.0       100     0.011         10.0       20        2000     0.0      0.0    ;   % 8
                         '19(C,U)-015P'    2.0       0.60            3.0              5.0            1E+06       0.0         15.0
45.0     75.0       100     0.011         10.0       20        2000     0.0      0.0    ;   % 9
                         '19(C,U)-030P'    2.0       0.60            3.0              5.0            1E+06       0.0         30.0
90.0    150.0       100     0.011         10.0       20        2000     0.0      0.0    ;   % 10
                         '19(C,U)-050P'    2.0       0.60            3.0              5.0            1E+06       0.0         50.0
150.0    250.0       100     0.011         10.0       20        2000     0.0      0.0    ;   % 11
                         '19(C,U)-100P'    2.0       0.60            3.0              5.0            1E+06       0.0        100.0
300.0    500.0       100     0.011         10.0       20        2000     0.0      0.0    ;   % 12
                         '19(C,U)-200P'    2.0       0.60            3.0              5.0            1E+06       0.0        200.0
600.0   1000.0       100     0.011         10.0       20        2000     0.0      0.0    ;   % 13
                         '19(C,U)-300P'    2.0       0.60            3.0              5.0            1E+06       0.0        300.0
900.0   1500.0       100     0.011         10.0       20        2000     0.0      0.0    ;   % 14
                         '19(C,U)-500P'    2.0       0.60            2.4              4.8            1E+06       0.0        500.0
1200.0   2400.0      100     0.011         10.0       20        2000     0.0      0.0    ;   % 15
                         'OMRON E8Y'       3.0       0.0             0.0             25.14              0         0.0          0.29
0.0      7.25       30      0.0          10.0       10         150     25.0     85.0    ;   % 16
                         'OMRON E8Y'       3.0       0.0             0.0              9.93              0         0.0          0.73
0.0      7.25       30      0.0          10.0       10         150     25.0     85.0    ;   % 17
                         '13(C,U)0500P'    2.0       0.60            3.0              5.0            1E+06       0.0        500.0
1500.0   2500.0      100     0.011         10.0       20        2000     0.0      0.0    ;   % 18
                         '13(C,U)1000P'    2.0       0.60            3.0              5.0            1E+06       0.0       1000.0
3000.0   5000.0      100     0.011         10.0       20        2000     0.0      0.0    ;   % 19
                         '13(C,U)2000P'    2.0       0.60            3.0              5.0            1E+06       0.0       2000.0
6000.0  10000.0      100     0.011         10.0       20        2000     0.0      0.0    ;   % 20
                         '13(C,U)3000P'    2.0       0.60            3.0             3.333           1E+06       0.0       3000.0
9000.0  10000.0      100     0.011         10.0       20        2000     0.0      0.0    ;   % 21
                         '13(C,U)5000P'    2.0       0.60            2.0              2.0            1E+06       0.0       5000.0
10000.0  10000.0      100     0.011         10.0       20        2000     0.0      0.0    ;   % 22
                         'PDCR 3500'       0.1       0.30            2.0              6.0              0         5.0       1000.0
2000.0   3000.0       0       0.0           0.0        0          0       0.0      0.0    ;   % 23
                         'PDCR 3500'       0.1       0.30            2.0              4.0              0         5.0      10000.0
10000.0  10000.0       0       0.0           0.0        0          0       0.0      0.0    ;   % 24
                         '2403SAT'        0.20       0.10            2.0              3.0              0         0.0       5000.0
10000.0  15000.0       30      0.011        25.0        0          0       0.0      0.0    ;   % 25
```

```
                    };

 % Properties       [    OPERATING TEMP   STORAGE TEMP     VOLTAGE   CURRENT   BANDWIDTH         MATERIAL
DIMENSIONS  (in)     WEIGHT       TEMPERATURE SENSOR PROPERTIES   ACCURACY    ]
 %    Units          [    Low deg F High  Low deg F High      V         mA          dB       SENSOR          CASE
Length  Width  Height     oz.          Low   SENSING RANGE  High        deg F       ]
    PRESSSENSOR_DB2 = {  -40.0    220.0  -60.0    250.0      5.0       5.0        -3.0   'Stainless Steel' 'Stainless Steel'
1.912  0.875  0.875       0.0           -65.0                250.0       2.0      ;   %  1
                        -40.0    220.0  -60.0    250.0      5.0       5.0        -3.0   'Stainless Steel' 'Stainless Steel'
1.912  0.875  0.875       0.0           -65.0                250.0       2.0      ;   %  2
                        -40.0    220.0  -60.0    250.0      5.0       5.0        -3.0   'Stainless Steel' 'Stainless Steel'
1.912  0.875  0.875       0.0           -65.0                250.0       2.0      ;   %  3
                        -40.0    220.0  -60.0    250.0      5.0       5.0        -3.0   'Stainless Steel' 'Stainless Steel'
1.912  0.875  0.875       0.0           -65.0                250.0       2.0      ;   %  4
                        -40.0    220.0  -60.0    250.0      5.0       5.0        -3.0   'Stainless Steel' 'Stainless Steel'
1.912  0.875  0.875       0.0           -65.0                250.0       2.0      ;   %  5
                        -40.0    257.0  -40.0    257.0      5.0       2.0         0     'Stainless Steel' 'Stainless Steel'
3.365  0.750  0.750       0.0             0                  0           0       ;   %  6
                        -40.0    257.0  -40.0    257.0      5.0       2.0         0     'Stainless Steel' 'Stainless Steel'
3.365  0.750  0.750       0.0             0                  0           0       ;   %  7
                        -40.0    257.0  -40.0    257.0      5.0       2.0         0     'Stainless Steel' 'Stainless Steel'
3.365  0.750  0.750       0.0             0                  0           0       ;   %  8
                        -40.0    257.0  -40.0    257.0      5.0       2.0         0     'Stainless Steel' 'Stainless Steel'
3.365  0.750  0.750       0.0             0                  0           0       ;   %  9
                        -40.0    257.0  -40.0    257.0      5.0       2.0         0     'Stainless Steel' 'Stainless Steel'
3.365  0.750  0.750       0.0             0                  0           0       ;   % 10
                        -40.0    257.0  -40.0    257.0      5.0       2.0         0     'Stainless Steel' 'Stainless Steel'
3.365  0.750  0.750       0.0             0                  0           0       ;   % 11
                        -40.0    257.0  -40.0    257.0      5.0       2.0         0     'Stainless Steel' 'Stainless Steel'
3.365  0.750  0.750       0.0             0                  0           0       ;   % 12
                        -40.0    257.0  -40.0    257.0      5.0       2.0         0     'Stainless Steel' 'Stainless Steel'
3.365  0.750  0.750       0.0             0                  0           0       ;   % 13
                        -40.0    257.0  -40.0    257.0      5.0       2.0         0     'Stainless Steel' 'Stainless Steel'
3.365  0.750  0.750       0.0             0                  0           0       ;   % 14
                        -40.0    257.0  -40.0    257.0      5.0       2.0         0     'Stainless Steel' 'Stainless Steel'
3.365  0.750  0.750       0.0             0                  0           0       ;   % 15
                         14.0    131.0  -13.0    149.0     30.0      50.0         0     'Resin pipe/Zinc'         'PBT'
1.220  1.180  1.180       2.8             0                  0           0       ;   % 16
                         14.0    131.0  -13.0    149.0     30.0      50.0         0     'Resin pipe/Zinc'         'PBT'
1.220  1.180  1.180       2.8             0                  0           0       ;   % 17
                        -40.0    257.0  -40.0    257.0     15.0       2.0         0     'Stainless Steel' 'Stainless Steel'
3.190  0.750  0.750       0.0             0                  0           0       ;   % 18
                        -40.0    257.0  -40.0    257.0     15.0       2.0         0     'Stainless Steel' 'Stainless Steel'
3.190  0.750  0.750       0.0             0                  0           0       ;   % 19
                        -40.0    257.0  -40.0    257.0     15.0       2.0         0     'Stainless Steel' 'Stainless Steel'
3.190  0.750  0.750       0.0             0                  0           0       ;   % 20
                        -40.0    257.0  -40.0    257.0     15.0       2.0         0     'Stainless Steel' 'Stainless Steel'
3.190  0.750  0.750       0.0             0                  0           0       ;   % 21
                        -40.0    257.0  -40.0    257.0     15.0       2.0         0     'Stainless Steel' 'Stainless Steel'
3.190  0.750  0.750       0.0             0                  0           0       ;   % 22
                        -65.0    250.0  -65.0    250.0      5.0       0.0         0     'Stainless Steel' 'Stainless Steel'
3.650  1.000  1.000       0.0             0                  0           0       ;   % 23
                        -65.0    250.0  -65.0    250.0     15.0       2.0         0     'Stainless Steel' 'Stainless Steel'
4.050  1.000  1.000       0.0             0                  0           0       ;   % 24
```

```
                        -65.0     250.0  -65.0     250.0      36.0       0.0            0   'Stainless Steel' 'Stainless Steel'
3.500  1.250  1.250      6.0             0                     0          0      ;   % 25
    };


    PressUnitLabelStr1 = {  'Type'   '%'   '% @ 1 yr'   'N/A'   'N/A'   'Cycles'   'psig'  'psig'  'psig'   'psig'   'g''s'
'sec'   'g''s'  'Hz'   'Hz' '% RH'  '% RH'   };
    PressUnitLabelStr2 = {  'deg F'  'deg F'  'deg F'  'deg F'  'V'   'mA'   'dB'   'N/A'   'N/A'   'in'   'in'   'in'   'oz'
'deg F'  'deg F'   'deg F'    };
end

%  Combine Pressure Sensor Arrays
PRESSSENSOR_DB = cat(2,PRESSSENSOR_DB1,PRESSSENSOR_DB2);
PressUnitLabelStr = cat(2,PressUnitLabelStr1,PressUnitLabelStr2);

%   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.' ;
    PRESSSENSOR_DB{25,34} = 'TABER Industries. Flight Proven.  Model extends 2403 design. Flown on ChinaStar, Clementine, EOS,
Landsat, Wakeshield, Seastar, IndoStar, Lockheed Astro A2100, 7000, NASA-JPL Deep Space One mission.' ;


%  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

end

if exist('P_MATRIX') == 0
    fprintf('\n\n  ***  WARNING  ***');
    fprintf('\n  NO VIABLE PRESSURE SENSOR EXISTS IN DATABASE!');
    fprintf('\n  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)
                P_MATRIX3{c,a} = P_MATRIX2{t,a};
            end
        c = c+1;
        end
    end

end

%  Copies P_MATRIX2 to P_MATRIX3 if all Sensors Eliminated
if exist('P_MATRIX3') == 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,8};
    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
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


%     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_MATRIX,2)
%                     P_MATRIX5{c,a} = P_MATRIX4{t,a};
%                 end
```

```
%             c = c+1;
%             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;
%     end

else
    P_MATRIX6 = P_MATRIX;
end


if exist('P_MATRIX') == 0
    for wv = 1:size(PRESSSENSOR_DB,2)
        P_MATRIX6{1,w} = 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('\n\n   ***   P R E S S U R E   S E N S O R   S U M M A R Y   R E S U L T S   ***');
    fprintf('\n\n***   I N P U T S   ***');
    fprintf('\n  Planetary Surface Pressure:   %7.2f  %s', PLPRESS, PressureUnit);
    fprintf('\n  Unit System:          %s', UNITS);

    fprintf('\n\n*************************************************************************\n');

    %  Pressure Sensor Properties
    fprintf('\n\n** Sensor Properties **');
    fprintf('\n  Pressure Sensor Type:                 %s', P_MATRIX6{1,1});
    fprintf('\n  Pressure Sensor Accuracy:          %10.3f   %s', P_MATRIX6{1,2}, PressUnitLabelStr{2} );
    fprintf('\n  Pressure Sensor Stability:         %10.3f   %s', P_MATRIX6{1,3}, PressUnitLabelStr{3} );
    fprintf('\n  Over Pressure Rated Value:         %10.3f   %s', P_MATRIX6{1,4}, PressUnitLabelStr{4} );
    fprintf('\n  Burst Pressure Rating:             %10.3f   %s', P_MATRIX6{1,5}, PressUnitLabelStr{5} );
    fprintf('\n  Life Cycles:                    %14.3E   %s', P_MATRIX6{1,6}, PressUnitLabelStr{6} );
    fprintf('\n  Sensing Pressure Range - Low:      %10.3f   %s', P_MATRIX6{1,7}, PressUnitLabelStr{7} );
    fprintf('\n                          High   :    %10.3f   %s', P_MATRIX6{1,8}, PressUnitLabelStr{8} );
    fprintf('\n  Proof Pressure:                    %10.3f   %s', P_MATRIX6{1,9}, PressUnitLabelStr{9} );
    fprintf('\n  Burst Pressure:                    %10.3f   %s', P_MATRIX6{1,10}, PressUnitLabelStr{10} );

    %  Environmental Properties
    fprintf('\n\n   Environmental Properties');
    fprintf('\n  Maximum Shock Load:                %10.3f   %s', P_MATRIX6{1,11}, PressUnitLabelStr{11} );
    fprintf('\n  Shock Impulse Time:                %10.3f   %s', P_MATRIX6{1,12}, PressUnitLabelStr{12} );
    fprintf('\n  Vibration Shock Limit:             %10.3f   %s', P_MATRIX6{1,13}, PressUnitLabelStr{13} );
    fprintf('\n  Operating Frequency Range - Low:   %10.3f   %s', P_MATRIX6{1,14}, PressUnitLabelStr{14} );
    fprintf('\n                              High:   %10.3f   %s', P_MATRIX6{1,15}, PressUnitLabelStr{15} );
    fprintf('\n  Operating Humidity Range - Low:    %10.3f   %s', P_MATRIX6{1,16}, PressUnitLabelStr{16} );
    fprintf('\n                             High:    %10.3f   %s', P_MATRIX6{1,17}, PressUnitLabelStr{17} );
    fprintf('\n  Operating Temperature Range - Low: %10.3f   %s', P_MATRIX6{1,18}, PressUnitLabelStr{18} );
    fprintf('\n                               High:  %10.3f   %s', P_MATRIX6{1,19}, PressUnitLabelStr{19} );
    fprintf('\n  Storage Temperature Range - Low:   %10.3f   %s', P_MATRIX6{1,20}, PressUnitLabelStr{20} );
    fprintf('\n                              High:   %10.3f   %s', P_MATRIX6{1,21}, PressUnitLabelStr{21} );

    fprintf('\n\n  Power Requirements');
    fprintf('\n  Input Voltage:                     %10.3f   %s', P_MATRIX6{1,22}, PressUnitLabelStr{22} );
    fprintf('\n  Input Amperage:                    %10.3f   %s', P_MATRIX6{1,23}, PressUnitLabelStr{23} );
    fprintf('\n  Sensing Bandwidth:                 %10.3f   %s', P_MATRIX6{1,24}, PressUnitLabelStr{24} );

    fprintf('\n\n  Physical Properties');
    fprintf('\n  Sensor Material:     %s', P_MATRIX6{1,25});
    fprintf('\n  Case Material:       %s', P_MATRIX6{1,26});
    fprintf('\n  Dimensions - Length:  %10.3f   %s', P_MATRIX6{1,27}, PressUnitLabelStr{27} );
    fprintf('\n               Width:  %10.3f   %s', P_MATRIX6{1,28}, PressUnitLabelStr{28} );
    fprintf('\n              Height:  %10.3f   %s', P_MATRIX6{1,29}, PressUnitLabelStr{29} );
    fprintf('\n  Sensor Mass:         %10.3f   %s', P_MATRIX6{1,30}, PressUnitLabelStr{30} );
```

```
    %  Temperature Sensing Range
    fprintf('\n\n  Temperature Sensing Range');
    fprintf('\n  Lower Sensing Limit:     %10.3f   %s', P_MATRIX6{1,31}, PressUnitLabelStr{31} );
    fprintf('\n  Upper Sensing Limit:     %10.3f   %s', P_MATRIX6{1,32}, PressUnitLabelStr{32} );
    fprintf('\n  Accuracy:                %10.3f   %s', P_MATRIX6{1,33}, PressUnitLabelStr{33} );

    fprintf('\n\n  Sensor Comments:');
    fprintf('\n  %s', P_MATRIX6{ 1,34});

end


%**************************************************************************
%***     Writes a text file summary of the Planetary Properties

    Fout = fopen('PRESSURE_SENSOR_Summary.txt','w+');

    fprintf(Fout, '\n\n   ***   P R E S S U R E   S E N S O R   S U M M A R Y   R E S U L T S   ***');
    fprintf(Fout, '\n\n***   I N P U T S   ***');
    fprintf(Fout, '\n  Planetary Surface Pressure:   %7.2f  %s', PLPRESS, PressureUnit);
    fprintf(Fout, '\n  Unit System:          %s', UNITS);

    fprintf(Fout, '\n\n*********************************************************************************\n');

    %  Pressure Sensor Properties
    fprintf(Fout, '\n\n** Sensor Properties **');
    fprintf(Fout, '\n  Pressure Sensor Type:                 %s', P_MATRIX6{1,1});
    fprintf(Fout, '\n  Pressure Sensor Accuracy:          %10.3f   %s', P_MATRIX6{1,2}, PressUnitLabelStr{2} );
    fprintf(Fout, '\n  Pressure Sensor Stability:         %10.3f   %s', P_MATRIX6{1,3}, PressUnitLabelStr{3} );
    fprintf(Fout, '\n  Over Pressure Rated Value:         %10.3f   %s', P_MATRIX6{1,4}, PressUnitLabelStr{4} );
    fprintf(Fout, '\n  Burst Pressure Rating:             %10.3f   %s', P_MATRIX6{1,5}, PressUnitLabelStr{5} );
    fprintf(Fout, '\n  Life Cycles:                    %14.3E   %s', P_MATRIX6{1,6}, PressUnitLabelStr{6} );
    fprintf(Fout, '\n  Sensing Pressure Range - Low:      %10.3f   %s', P_MATRIX6{1,7}, PressUnitLabelStr{7} );
    fprintf(Fout, '\n                          High   :   %10.3f   %s', P_MATRIX6{1,8}, PressUnitLabelStr{8} );
    fprintf(Fout, '\n  Proof Pressure:                    %10.3f   %s', P_MATRIX6{1,9}, PressUnitLabelStr{9} );
    fprintf(Fout, '\n  Burst Pressure:                    %10.3f   %s', P_MATRIX6{1,10}, PressUnitLabelStr{10} );

    %  Environmental Properties
    fprintf(Fout, '\n\n   Environmental Properties');
    fprintf(Fout, '\n  Maximum Shock Load:                %10.3f   %s', P_MATRIX6{1,11}, PressUnitLabelStr{11} );
    fprintf(Fout, '\n  Shock Impulse Time:                %10.3f   %s', P_MATRIX6{1,12}, PressUnitLabelStr{12} );
    fprintf(Fout, '\n  Vibration Shock Limit:             %10.3f   %s', P_MATRIX6{1,13}, PressUnitLabelStr{13} );
    fprintf(Fout, '\n  Operating Frequency Range - Low:   %10.3f   %s', P_MATRIX6{1,14}, PressUnitLabelStr{14} );
    fprintf(Fout, '\n                              High:   %10.3f   %s', P_MATRIX6{1,15}, PressUnitLabelStr{15} );
    fprintf(Fout, '\n  Operating Humidity Range - Low:    %10.3f   %s', P_MATRIX6{1,16}, PressUnitLabelStr{16} );
    fprintf(Fout, '\n                             High:   %10.3f   %s', P_MATRIX6{1,17}, PressUnitLabelStr{17} );
    fprintf(Fout, '\n  Operating Temperature Range - Low: %10.3f   %s', P_MATRIX6{1,18}, PressUnitLabelStr{18} );
    fprintf(Fout, '\n                               High: %10.3f   %s', P_MATRIX6{1,19}, PressUnitLabelStr{19} );
    fprintf(Fout, '\n  Storage Temperature Range - Low:   %10.3f   %s', P_MATRIX6{1,20}, PressUnitLabelStr{20} );
    fprintf(Fout, '\n                              High:   %10.3f   %s', P_MATRIX6{1,21}, PressUnitLabelStr{21} );

    fprintf(Fout, '\n\n  Power Requirements');
```

```
fprintf(Fout, '\n  Input Voltage:                    %10.3f   %s', P_MATRIX6{1,22}, PressUnitLabelStr{22} );
fprintf(Fout, '\n  Input Amperage:                   %10.3f   %s', P_MATRIX6{1,23}, PressUnitLabelStr{23} );
fprintf(Fout, '\n  Sensing Bandwidth:                %10.3f   %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:  %10.3f   %s', P_MATRIX6{1,27}, PressUnitLabelStr{27} );
fprintf(Fout, '\n               Width:  %10.3f   %s', P_MATRIX6{1,28}, PressUnitLabelStr{28} );
fprintf(Fout, '\n              Height:  %10.3f   %s', P_MATRIX6{1,29}, PressUnitLabelStr{29} );
fprintf(Fout, '\n  Sensor Mass:         %10.3f   %s', P_MATRIX6{1,30}, PressUnitLabelStr{30} );


%  Temperature Sensing Range
fprintf(Fout, '\n\n  Temperature Sensing Range');
fprintf(Fout, '\n  Lower Sensing Limit:     %10.3f   %s', P_MATRIX6{1,31}, PressUnitLabelStr{31} );
fprintf(Fout, '\n  Upper Sensing Limit:     %10.3f   %s', P_MATRIX6{1,32}, PressUnitLabelStr{32} );
fprintf(Fout, '\n  Accuracy:                %10.3f   %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-detector.shtml
%  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
```

```
%  http://www.amptek.com/x123.html
%  http://polimaster.us/products/searching/pm1401gn-gna-gnb.php

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
 %  Properties          [    SENSOR                   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 RANGE     X-RAY BINS    PARTICLE COUNT       G-FACTOR        WEIGHT
DIMENSIONS - mm       POWER    ]
 %    Units             [     TYPE                         N/A                    Low    MeV    High      # of Bins      Low    MeV
High         # of Bins      Low    MeV    High       # of Bins      Low   MeV   High       # of Bins      Low    keV    High
# of Bins    Low  keV  High    # of Bins      Low  keV  High    # of Bins          kcps              cm^2 sr          kg       Len
Wid     Hgt        W      ]
   RADIATIONSENSOR_DB =  {     'LPD'             { 'Charged Particle' }       1.00         250.00         15         0.50
20.00            7          6.00          250.00          6         1.50          0             1          0              0
0        0         0          0           0         0         0             100      [ 0.05 0.1 0.3 ]    7.00      150    150
300     15.0    ;
                              'LPD-HECPS'        { 'Charged Particle' }       1.00         150.00         5          0.30
10.00            5         12.00           0              0         3.00          0             1          0              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        3.00          0              1         0             0             0         50.00          0
1       7.00       0          1          7.00       0         1              0             [ 0 ]        0.264      135     86
38       0    ;
                               'M4EC'      { 'Alpha' 'Beta' 'Gamma' 'X-Ray' }  0            0             0          0
0               0        2.50          0              1         0             0             0         50.00          0
1      10.00    1300.0        1         10.00   1300.0         1              0             [ 0 ]        0.246      210     70
48       0    ;
                               'ASM-993'       { 'Alpha' 'Beta' 'Gamma' }      0            0             0          0
0               0        3.50          0              1         0             0             0         35.00          0
1       6.00       0          1           0         0         0             83.333         [ 0 ]        1.09       277.1
104.7    63.5      0    ;
                          'CAPture Plus'               { 'Gamma' }             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         15     15
15       0    ;
                               'X-123'                 { 'X-Ray' }             0            0             0          0
0               0          0           0              0         0             0             0           0            0
0        0         0          0          1.00      40.00        1             200            [ 0 ]        0.180       70    100
25      1.2    ;
                              'PM1401GNB'    { 'Charged Particle' 'Gamma' }    0            0             0          0
0               0          0           0              0       2.5E-8        14.0            1           0            0
```

```
0       33.00     3000.0         1           0          0          0              0              [ 0 ]          0.450      195      57
32        0     };

    RadUnitLabelStr = { 'N/A'   'N/A'   'MeV'   'MeV'   '# of Bins'   'MeV'   'MeV'   '# of Bins'   'MeV'   'MeV'   '# of Bins'   'MeV'
'MeV'   '# of Bins'   'keV'   'keV'   '# of Bins'   'keV'   'keV'   '# of Bins'   'keV'   'keV'   '# of Bins'   'kcps'   'cm^2 sr'   'kg'
'mm'   'mm'   'mm'   'W'   };

end


if strcmpi(UNITS,'British') == 1
 %  Properties          [    SENSOR                  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 RANGE     X-RAY BINS     PARTICLE COUNT      G-FACTOR       WEIGHT
DIMENSIONS - in       POWER   ]
 %    Units              [     TYPE                        N/A                  Low    MeV    High      # of Bins      Low    MeV
High        # of Bins      Low    MeV    High       # of Bins      Low   MeV   High       # of Bins      Low    keV    High
# of Bins    Low  keV  High    # of Bins     Low  keV  High    # of Bins          kcps        x10^-3in^2 sr      oz       Len
Wid     Hgt    BTU/hr  ]
   RADIATIONSENSOR_DB =  {    'LPD'            { 'Charged Particle' }       1.00         250.00          15          0.50
20.00              7          6.00         250.00            6          1.50          0              1          0              0
0         0          0          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
10.00              5          12.00          0            0          3.00          0              1          0              0
0         0          0          0           0          0          0             200           [ 31.0 ]       105.82     5.91   5.91
4.73    6.824  ;
                            'GM-180'   { 'Alpha' 'Beta' 'Gamma' 'X-Ray' }   0              0          0          0
0                 0         3.00           0            1          0             0              0          50.00          0
1       7.00        0          1          7.00         0          1             0              [ 0 ]          9.30     5.30   3.40
1.50      0     ;
                            'M4EC'     { 'Alpha' 'Beta' 'Gamma' 'X-Ray' }   0              0          0          0
0                 0         2.50           0            1          0             0              0          50.00          0
1      10.00     1300.0        1          10.00   1300.0       1             0              [ 0 ]          8.70     8.25   2.75
1.875     0     ;
                          'ASM-993'       { 'Alpha' 'Beta' 'Gamma' }       0              0          0          0
0                 0         3.50           0            1          0             0              0          35.00          0
1       6.00        0          1           0          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       30.00     1300.0        1           0          0          0             0              [ 0 ]          0        0.60   0.60
0.60      0     ;
                           'X-123'                 { 'X-Ray' }              0              0          0          0
0                 0           0            0            0          0             0              0            0            0
0         0          0          0           1.00     40.00        1            200             [ 0 ]          6.30     2.70   3.90
1.00    4.094  ;
                          'PM1401GNB'    { 'Charged Particle' 'Gamma' }     0              0          0          0
0                 0           0            0            0        2.5E-8         14.0             1          0            0
0       33.00     3000.0        1           0          0          0             0              [ 0 ]          15.90    7.70   2.20
1.30      0     };
```

```
    RadUnitLabelStr = { 'N/A'  'N/A'  'MeV'  'MeV'  '# of Bins'  'MeV'  'MeV'  '# of Bins'  'MeV'  'MeV'  '# of Bins'  'MeV'
'MeV'  '# of Bins'  'keV'  'keV'  '# of Bins'  'keV'  'keV'  '# of Bins'  'keV'  'keV'  '# of Bins''kcps'  'x10^-3in^2 sr'
'oz'  'in'  'in'  'in'  'BTUs/hr'  };

end


% 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: DRTS-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)'   ;
                             'Available with radhard or COTS components'  ;
                             'Available as single-sided or fully redundant system'   };
RADIATIONSENSOR_DB{3,31} = { '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{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)'   ;
                             'Handheld Unit - USB option'  };
RADIATIONSENSOR_DB{5,31} = { 'Internal probe properties given, Multiple external probe configurations available'  ;
                             'Three Data Modes - Manual, Timed, Survey'  ;
                             'Power Requirement: 2 D cell battery 150hrs lifetime'      ;
                             'Accuracy - ±10% of reading of full scale on any range'    };
RADIATIONSENSOR_DB{6,31} = { 'CZT Gamma-Ray Spectrometer'  ;
                             'Energy Resolution: <5% FWHM @ 122keV <3.0% @ 662keV'  ;
                             'Power Requirement: AC or DC coupled, w/ +/- 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 +5 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'   ;
                             'complete X-Ray Detector System'  ;
                             'Typical Device Lifetime 5 to 10 years, depending on use'    };
RADIATIONSENSOR_DB{8,31} = { 'Detector Type - Gamma Ray and Neutron'  ;
                             'Detects and identifies Special Nuclear Materials, Naturally Occurring Radioactive
Materials(NORM), and Industrial radionucleotides'  ;
                             'Power Requirement: One AA Battery 800hr life'      ;
                             'Communications - PC Communication mode (IR, Bluetooth)'   ;
```

```
                            'Identifies: 233U, 235U, 237Np, Pu, 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
             if FLAG == 1
               c = c + 1;
            end
            FLAG = 0;
         end

         if exist('RAD_MATRIX_TMP') == 0
             if strcmpi(Print,'Y') == 1
                 fprintf('\n\n   ***  ERROR:  NULL DATABASE  ***');
                 fprintf('\n  No Sensor Exist in Database which Meet Radiation Requirements!');
                 fprintf('\n  Radiation Type:    %s', STR1);
                 RAD_MATRIX_TMP = RAD_MATRIX;
             end
         end

         clear RAD_MATRIX;
         RAD_MATRIX = RAD_MATRIX_TMP;
         clear RAD_MATRIX_TMP;
     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
```

```
                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('\n\n   ***  ERROR:  NULL DATABASE  ***');
            fprintf('\n  No Sensor Exist in Database which Meet Radiation Requirements!');
            fprintf('\n  Image Type:    %s', STR1);
            RAD_MATRIX_TMP = RAD_MATRIX;
        end
    end

    clear RAD_MATRIX;
    RAD_MATRIX = RAD_MATRIX_TMP;
    clear RAD_MATRIX_TMP;
end

if strcmpi(STR1, 'Beta') == 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(v,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

    if exist('RAD_MATRIX_TMP') == 0
        if strcmpi(Print,'Y') == 1
            fprintf('\n\n   ***  ERROR:  NULL DATABASE  ***');
            fprintf('\n  No Sensor Exist in Database which Meet Radiation Requirements!');
            fprintf('\n  Image Type:    %s', STR1);
            RAD_MATRIX_TMP = RAD_MATRIX;
        end
    end

    clear RAD_MATRIX;
    RAD_MATRIX = RAD_MATRIX_TMP;
    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} = RAD_MATRIX{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('\n\n    ***  ERROR:  NULL DATABASE  ***');
            fprintf('\n  No Sensor Exist in Database which Meet Radiation Requirements!');
            fprintf('\n  Image Type:    %s', STR1);
            RAD_MATRIX_TMP = RAD_MATRIX;
        end
    end

    clear RAD_MATRIX;
    RAD_MATRIX = RAD_MATRIX_TMP;
    clear RAD_MATRIX_TMP;
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
```

```
        if exist('RAD_MATRIX_TMP') == 0
            if strcmpi(Print,'Y') == 1
                fprintf('\n\n   ***  ERROR:  NULL DATABASE  ***');
                fprintf('\n  No Sensor Exist in Database which Meet Radiation Requirements!');
                fprintf('\n  Image Type:    %s', STR1);
                RAD_MATRIX_TMP = RAD_MATRIX;
            end
        end

        clear RAD_MATRIX;
        RAD_MATRIX = RAD_MATRIX_TMP;
        clear RAD_MATRIX_TMP;
    end

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

            %  Add up Heavy Nucleon Energy Range
            for n = 1:size(RAD_MATRIX,1)
                SUM_RAD_ARRAY(n) = SUM_RAD_ARRAY(n) + RAD_MATRIX{n,13} - RAD_MATRIX{n,12};
            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

        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} - RAD_MATRIX{b,15};
            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} - RAD_MATRIX{g,18};
            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) + RAD_MATRIX{a,22} - RAD_MATRIX{a,21};
            end
        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
        %  with the Greatest Range
        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

end

if exist('RAD_MATRIX2') == 0
    RAD_MATRIX2 = RAD_MATRIX;
end


if size(RAD_MATRIX2,1) > 1
    %  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};

        if (RAD_MATRIX2{v,27} == 0) || (RAD_MATRIX2{v,28} == 0) || (RAD_MATRIX2{v,29} == 0)
```

```
            RAD_SCALE_FAC(v) = 9.99E99;
        end
    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

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('\n\n   ***   R A D I A T I O N   S E N S O R   S U M M A R Y   R E S U L T S   ***');
    fprintf('\n\n***   I N P U T S   ***');
    fprintf('\n  Unit System:          %s', UNITS);

    fprintf('\n\n*************************************************************************\n');

    %  Radiation Sensor Properties
    fprintf('\n\n** Sensor Properties **');
    fprintf('\n  Radiation Sensor Type:           %s', RAD_MATRIX3{1,1});
    fprintf('\n  Radiation Detected:                ');
    for r = 1:length(RAD_MATRIX3{1,2})
        fprintf('    %s', RAD_MATRIX3{1,2}{r} );
    end
    fprintf('\n  Proton Energy Range - Low:      %11.3f   %s', RAD_MATRIX3{1,3}, RadUnitLabelStr{3} );
    fprintf('\n                       High:      %11.3f   %s', RAD_MATRIX3{1,4}, RadUnitLabelStr{4} );
    fprintf('\n  Number of Detection Bins:       %11d   %s', RAD_MATRIX3{1,5}, RadUnitLabelStr{5} );
    fprintf('\n  Electron Energy Range - Low:    %11.3f   %s', RAD_MATRIX3{1,6}, RadUnitLabelStr{6} );
    fprintf('\n                         High:    %11.3f   %s', RAD_MATRIX3{1,7}, RadUnitLabelStr{7} );
```

```
    fprintf('\n  Number of Detection Bins:       %11d   %s', RAD_MATRIX3{1,8}, RadUnitLabelStr{8} );
    fprintf('\n  Alpha Particle Range - Low:     %11.3f   %s', RAD_MATRIX3{1,9}, RadUnitLabelStr{9} );
    fprintf('\n                         High:    %11.3f   %s', RAD_MATRIX3{1,10}, RadUnitLabelStr{10} );
    fprintf('\n  Number of Detection Bins:       %11d   %s', RAD_MATRIX3{1,11}, RadUnitLabelStr{11} );
    fprintf('\n  Nucleon/Heavy Ion Range - Low:  %11.3f   %s', RAD_MATRIX3{1,12}, RadUnitLabelStr{12} );
    fprintf('\n                          High:   %11.3f   %s', RAD_MATRIX3{1,13}, RadUnitLabelStr{13} );
    fprintf('\n  Number of Detection Bins:       %11d   %s', RAD_MATRIX3{1,14}, RadUnitLabelStr{14} );
    fprintf('\n  Beta Particle Range - Low:      %11.3f   %s', RAD_MATRIX3{1,15}, RadUnitLabelStr{15} );
    fprintf('\n                         High:    %11.3f   %s', RAD_MATRIX3{1,16}, RadUnitLabelStr{16} );
    fprintf('\n  Number of Detection Bins:       %11d   %s', RAD_MATRIX3{1,17}, RadUnitLabelStr{17} );
    fprintf('\n  Gamma Particle Range - Low:     %11.3f   %s', RAD_MATRIX3{1,18}, RadUnitLabelStr{18} );
    fprintf('\n                         High:    %11.3f   %s', RAD_MATRIX3{1,19}, RadUnitLabelStr{19} );
    fprintf('\n  Number of Detection Bins:       %11d   %s', RAD_MATRIX3{1,20}, RadUnitLabelStr{20} );
    fprintf('\n  X-Ray Energy Range - Low:       %11.3f   %s', RAD_MATRIX3{1,21}, RadUnitLabelStr{21} );
    fprintf('\n                        High:     %11.3f   %s', RAD_MATRIX3{1,22}, RadUnitLabelStr{22} );
    fprintf('\n  Number of Detection Bins:       %11d   %s', RAD_MATRIX3{1,23}, RadUnitLabelStr{23} );
    fprintf('\n  Particle Detection Rate:        %11.3f   %s', RAD_MATRIX3{1,24}, RadUnitLabelStr{24} );
    fprintf('\n  G-Factor:                    ');
    for g = 1:length(RAD_MATRIX3{1,25})
        fprintf('    %6.3f', RAD_MATRIX3{1,25}(g) );
    end
    fprintf('    %s', RadUnitLabelStr{25} );


    fprintf('\n\n  Physical Properties');
    fprintf('\n  Sensor Mass:           %11.3f   %s', RAD_MATRIX3{1,26}, RadUnitLabelStr{26} );
    fprintf('\n  Dimensions - Length:   %11.3f   %s', RAD_MATRIX3{1,27}, RadUnitLabelStr{27} );
    fprintf('\n                Width:   %11.3f   %s', RAD_MATRIX3{1,28}, RadUnitLabelStr{28} );
    fprintf('\n               Height:   %11.3f   %s', RAD_MATRIX3{1,29}, RadUnitLabelStr{29} );
    fprintf('\n  Power Requirement:     %11.3f   %s', RAD_MATRIX3{1,30}, RadUnitLabelStr{30} );


    fprintf('\n\n  Sensor Comments:');
    for r = 1:length(RAD_MATRIX3{1,31})
        fprintf('\n    %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, '\n\n   ***   R A D I A T I O N   S E N S O R   S U M M A R Y   R E S U L T S   ***');
    fprintf(Fout, '\n\n***   I N P U T S   ***');
    fprintf(Fout, '\n  Unit System:           %s', UNITS);

    fprintf(Fout, '\n\n*************************************************************************\n');

    %  Radiation Sensor Properties
    fprintf(Fout, '\n\n** Sensor Properties **');
    fprintf(Fout, '\n  Radiation Sensor Type:           %s', RAD_MATRIX3{1,1});
```

```
fprintf(Fout, '\n  Radiation Detected:                    ');
for r = 1:length(RAD_MATRIX3{1,2})
    fprintf(Fout, '    %s', RAD_MATRIX3{1,2}{r} );
end
fprintf(Fout, '\n  Proton Energy Range - Low:      %11.3f   %s', RAD_MATRIX3{1,3}, RadUnitLabelStr{3} );
fprintf(Fout, '\n                        High:      %11.3f   %s', RAD_MATRIX3{1,4}, RadUnitLabelStr{4} );
fprintf(Fout, '\n  Number of Detection Bins:        %11d   %s', RAD_MATRIX3{1,5}, RadUnitLabelStr{5} );
fprintf(Fout, '\n  Electron Energy Range - Low:     %11.3f   %s', RAD_MATRIX3{1,6}, RadUnitLabelStr{6} );
fprintf(Fout, '\n                         High:     %11.3f   %s', RAD_MATRIX3{1,7}, RadUnitLabelStr{7} );
fprintf(Fout, '\n  Number of Detection Bins:        %11d   %s', RAD_MATRIX3{1,8}, RadUnitLabelStr{8} );
fprintf(Fout, '\n  Alpha Particle Range - Low:      %11.3f   %s', RAD_MATRIX3{1,9}, RadUnitLabelStr{9} );
fprintf(Fout, '\n                         High:     %11.3f   %s', RAD_MATRIX3{1,10}, RadUnitLabelStr{10} );
fprintf(Fout, '\n  Number of Detection Bins:        %11d   %s', RAD_MATRIX3{1,11}, RadUnitLabelStr{11} );
fprintf(Fout, '\n  Nucleon/Heavy Ion Range - Low:   %11.3f   %s', RAD_MATRIX3{1,12}, RadUnitLabelStr{12} );
fprintf(Fout, '\n                           High:   %11.3f   %s', RAD_MATRIX3{1,13}, RadUnitLabelStr{13} );
fprintf(Fout, '\n  Number of Detection Bins:        %11d   %s', RAD_MATRIX3{1,14}, RadUnitLabelStr{14} );
fprintf(Fout, '\n  Beta Particle Range - Low:       %11.3f   %s', RAD_MATRIX3{1,15}, RadUnitLabelStr{15} );
fprintf(Fout, '\n                        High:      %11.3f   %s', RAD_MATRIX3{1,16}, RadUnitLabelStr{16} );
fprintf(Fout, '\n  Number of Detection Bins:        %11d   %s', RAD_MATRIX3{1,17}, RadUnitLabelStr{17} );
fprintf(Fout, '\n  Gamma Particle Range - Low:      %11.3f   %s', RAD_MATRIX3{1,18}, RadUnitLabelStr{18} );
fprintf(Fout, '\n                         High:     %11.3f   %s', RADIATIONSENSOR_DB{1,19}, RadUnitLabelStr{19} );
fprintf(Fout, '\n  Number of Detection Bins:        %11d   %s', RAD_MATRIX3{1,20}, RadUnitLabelStr{20} );
fprintf(Fout, '\n  X-Ray Energy Range - Low:        %11.3f   %s', RAD_MATRIX3{1,21}, RadUnitLabelStr{21} );
fprintf(Fout, '\n                        High:      %11.3f   %s', RAD_MATRIX3{1,22}, RadUnitLabelStr{22} );
fprintf(Fout, '\n  Number of Detection Bins:        %11d   %s', RAD_MATRIX3{1,23}, RadUnitLabelStr{23} );
fprintf(Fout, '\n  Particle Detection Rate:         %11.3f   %s', RAD_MATRIX3{1,24}, RadUnitLabelStr{24} );
fprintf(Fout, '\n  G-Factor:                    ');
for g = 1:length(RAD_MATRIX3{1,25})
    fprintf(Fout, '    %6.3f', RAD_MATRIX3{1,25}(g) );
end
fprintf(Fout, '    %s', RadUnitLabelStr{25} );


fprintf(Fout, '\n\n  Physical Properties');
fprintf(Fout, '\n  Sensor Mass:           %11.3f   %s', RAD_MATRIX3{1,26}, RadUnitLabelStr{26} );
fprintf(Fout, '\n  Dimensions - Length:   %11.3f   %s', RAD_MATRIX3{1,27}, RadUnitLabelStr{27} );
fprintf(Fout, '\n               Width:    %11.3f   %s', RAD_MATRIX3{1,28}, RadUnitLabelStr{28} );
fprintf(Fout, '\n              Height:    %11.3f   %s', RAD_MATRIX3{1,29}, RadUnitLabelStr{29} );
fprintf(Fout, '\n  Power Requirement:     %11.3f   %s', RAD_MATRIX3{1,30}, RadUnitLabelStr{30} );


fprintf(Fout, '\n\n  Sensor Comments:');
for r = 1:length(RAD_MATRIX3{1,31})
    fprintf(Fout, '\n    %s', RAD_MATRIX3{1,31}{r} );
end

fclose(Fout);
```

D16. Temperature Sensors

```
%   function TEMP_SENSORS(TEMP_SENS_TYPE,PLTEMP,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_pdfs/4025_Sensor_146MD.pdf
%   http://www.rdfcorp.com/products/aerospace/aerospace.shtml

function TEMP_SENSORS(TEMP_SENS_TYPE,PLTEMP,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  TEMPSENSOR_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     EMF over Max TRange     OPERATING  PLANETS USED  ]
 %    Units              TYPE        deg C          deg C        W - TBD      g - TBD    Length   Width    Height      Pascals -
TBD  %          deg C    LEVEL    + Lead       - Lead         mV          mV           Type         N/A         ]
    TEMPSENSOR_DB = {       'B'          0.000       1700.000        0.000        0.000        0.00      0.00      0.00        0.000
0.5         0.0         0    'Pt-30%Rh'   'Pt-6%Rh'      0.000     13.280       'Voltage'      [   ]       ;
                          'C'          0.000       2320.000        0.000        0.000        0.00      0.00      0.00        0.000
1.0         4.5         0     'W-5%Re'   'W-26%Re'      0.000     37.066       'Voltage'      [   ]       ;
```

```
                        'D'        0.000      2320.000       0.000       0.000       0.00      0.00      0.00        0.000
1.0        4.5        0     'W-3%Re'   'W-25%Re'      0.000     39.506       'Voltage'      [   ]       ;
                           'E'      -200.000       900.000       0.000       0.000       0.00      0.00      0.00
0.000        0.5        1.7        0     'Ni-Cr'    'Cu-Ni'       -9.835     76.373       'Voltage'      [   ]       ;
                           'G'        0.000      2320.000       0.000       0.000       0.00      0.00      0.00
0.000        1.0        4.5        0     'W'        'W-26%Re'      0.000     38.564       'Voltage'      [   ]       ;
                           'J'        0.000       750.000       0.000       0.000       0.00      0.00      0.00
0.000        0.75       2.2        0     'Fe'       'Cu-Ni'       -8.095     69.553       'Voltage'      [   ]       ;
                           'K'      -200.000      1250.000       0.000       0.000       0.00      0.00      0.00
0.000        0.75       2.2        0     'Ni-Cr'    'Ni-Al'       -6.458     54.886       'Voltage'      [   ]       ;
                           'N'      -270.000      1300.000       0.000       0.000       0.00      0.00      0.00
0.000        0.75       2.2        0   'Ni-Cr-Si'   'Ni-Si-Mg'    -4.345     47.513       'Voltage'      [   ]       ;
                           'R'        0.000      1450.000       0.000       0.000       0.00      0.00      0.00
0.000        0.25       1.5        0   'Pt-13%Rh'   'Pt'          -0.226     21.101       'Voltage'      [   ]       ;
                           'S'        0.000      1450.000       0.000       0.000       0.00      0.00      0.00
0.000        0.25       1.5        0   'Pt-10%Rh'   'Pt'          -0.236     18.693       'Voltage'      [   ]       ;
                           'T'      -250.000       350.000       0.000       0.000       0.00      0.00      0.00
0.000        0.75       1.0        0     'Cu'       'Cu-Ni'       -6.258     20.872       'Voltage'      [   ]       ;
                      'PT-102'    -259.150       599.850       0.0001      0.250       31.115    2.007     2.007       0.000
0.56       4.5        0    'Platinum'   'Platinum'     0.000       0.000     'Resistance'    [   ]       ;
                      'M-0146MD'  -269.000       400.000       0.0600      0.350       45.720    1.524     1.524       0.000
0.10       0.26       0    'Platinum'   'Platinum'     0.000       0.000     'Resistance'    [   ]       };

    TempUnitLabelStr = {  'Type'   'deg C'   'deg C'   'W'   'g'   'mm'   'mm'   'mm'   'Pascals'      '%'      'deg C'   'TRL'
'+Lead'   '-Lead'    'mV'   'mV'   'N/A'   };

end

if strcmpi(UNITS,'British') == 1
 %  Properties      [    SENSOR    COLD LIMIT    HOT LIMIT    POWER REQ    MASS REQ         DIMENSIONS  (in)      PRESSURE
ERROR TOL  ERROR TOL    TRL      Alloy Combination      EMF over Max TRange    OPERATING  PLANETS USED  ]
 %    Units                TYPE       deg F        deg F      BTU/sec - TBD  oz - TBD    Length  Width  Height      psi - TBD
%         deg F     LEVEL    + Lead       - Lead       mV          mV          Type        N/A       ]
    TEMPSENSOR_DB = {       'B'        32.000      3092.000       0.000       0.000        0.00    0.00     0.00       0.000
0.5        0.0        0    'Pt-30%Rh'   'Pt-6%Rh'      0.000     13.280       'Voltage'      [   ]       ;
                           'C'        32.000      4208.000       0.000       0.000        0.00    0.00     0.00       0.000
1.0        40.1       0     'W-5%Re'   'W-26%Re'      0.000     37.066       'Voltage'      [   ]       ;
                           'D'        32.000      4208.000       0.000       0.000        0.00    0.00     0.00       0.000
1.0        40.1       0     'W-3%Re'   'W-25%Re'      0.000     39.506       'Voltage'      [   ]       ;
                           'E'      -328.000      1652.000       0.000       0.000        0.00      0.00      0.00
0.000        0.5        35.06      0     'Ni-Cr'    'Cu-Ni'       -9.835     76.373       'Voltage'      [   ]       ;
                           'G'        32.000      4208.000       0.000       0.000        0.00      0.00      0.00
0.000        1.0        40.1       0     'W'        'W-26%Re'      0.000     38.564       'Voltage'      [   ]       ;
                           'J'        32.000      1382.000       0.000       0.000        0.00      0.00      0.00
0.000        0.75       35.96      0     'Fe'       'Cu-Ni'       -8.095     69.553       'Voltage'      [   ]       ;
                           'K'      -328.000      2282.000       0.000       0.000        0.00      0.00      0.00
0.000        0.75       35.96      0     'Ni-Cr'    'Ni-Al'       -6.458     54.886       'Voltage'      [   ]       ;
                           'N'      -450.000      2372.000       0.000       0.000        0.00      0.00      0.00
0.000        0.75       35.96      0   'Ni-Cr-Si'   'Ni-Si-Mg'    -4.345     47.513       'Voltage'      [   ]       ;
                           'R'        32.000      2642.000       0.000       0.000        0.00      0.00      0.00
0.000        0.25       34.7       0   'Pt-13%Rh'   'Pt'          -0.226     21.101       'Voltage'      [   ]       ;
                           'S'        32.000      2642.000       0.000       0.000        0.00      0.00      0.00
0.000        0.25       34.7       0   'Pt-10%Rh'   'Pt'          -0.236     18.693       'Voltage'      [   ]       ;
```

```
                                        'T'      -418.000        662.000        0.000        0.000          0.00   0.00    0.00
0.000        0.75       33.8         0     'Cu'       'Cu-Ni'       -6.258     20.872       'Voltage'      [   ]       ;
                         'PT-102'   -259.150       599.850      0.0001       0.0088       1.225  0.079   0.079     0.000
0.56       8.10       0    'Platinum'  'Platinum'     0.000       0.000      'Resistance'    [   ]       ;
                        'M-0146MD'   -269.000       400.000      0.0600       0.0123       1.80    0.060  0.060     0.000
0.10       0.47       0    'Platinum'  'Platinum'     0.000       0.000      'Resistance'    [   ]       };

    TempUnitLabelStr = {  'Type'   'deg F'   'deg F'   'BTU/sec'   'oz'   'in'   'in'   'in'   'psi'    '%'     'deg F'   'TRL'
'+Lead'  '-Lead'    'mV'    'mV'   'N/A'   };
end

%   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{ 2,19} = 'Vacuum, Inert, Hydrogen. Beware of Embrittlement. Not Practical Below 399°C (750°F). Not for
Oxidizing Atmosphere.' ;
    TEMPSENSOR_DB{ 3,19} = 'Vacuum, Inert, Hydrogen. Beware of Embrittlement. Not Practical Below 399°C (750°F). Not for
Oxidizing Atmosphere.' ;
    TEMPSENSOR_DB{ 4,19} = 'Oxidizing or Inert. Limited Use in Vacuum or Reducing. Highest EMF Change Per Degree.' ;
    TEMPSENSOR_DB{ 5,19} = 'Vacuum, Inert, Hydrogen. Beware of Embrittlement. Not Practical Below 399°C (750°F). Not for
Oxidizing Atmosphere.' ;
    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 strcmpi(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      1990.000     ;
                        '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                              Resistance To:
Material Construction          ]
%                [   Resistance                  Submersion       Solvent         Acid         Base           Flame
Humidity     Overall          Conductors       ]
    TCWIns_DB2 = {  'Good'          'Excellent'   'Good'          'Fair'        'Good'       'Good'        'Good'        'Good'
'PVC'              'PVC'            ;%   'PP' Type
                    'Excellent'     'Good'        'Excellent'     'Excellent'   'Excellent'  'Excellent'   'Excellent'
'Excellent'   'FEP Teflon'      'FEP Teflon'     ;%   'FF' Type
                    'Excellent'     'Good'        'Excellent'     'Excellent'   'Excellent'  'Excellent'   'Excellent'
'Excellent'   'PFA Teflon'      'PFA Teflon'     ;%   'TT' Type
                    'Excellent'     'Good'        'Good'          'Good'        'Good'       'Good'        'Good'
'Excellent'   'Kapton'          'Kapton'         ;%   'KK' Type
                    'Good'          'Good'        'Excellent'     'Excellent'   'Excellent'  'Excellent'   'Excellent'
'Excellent'   'Glass Braid'     'PFA Teflon'     ;%   'TG' Type
                    'Poor'          'Good'        'Poor'          'Excellent'   'Excellent'  'Excellent'   'Excellent'   'Fair'
'Glass Braid'     'Glass Braid'    ;%   'GG' Type
                    'Poor'          'Good'        'Poor'          'Excellent'   'Excellent'  'Excellent'   'Excellent'   'Fair'
'HT Glass Braid'  'HT Glass Braid' ;%   'HH' Type
                    'Poor'          'Good'        'Poor'          'Excellent'   'Good'       'Good'        'Excellent'   'Poor'
'Refrasil Braid'  'Refrasil Braid' ;%   'XR' Type
                    'Poor'          'Good'        'Poor'          'Excellent'   'Good'       'Good'        'Excellent'   'Fair'
'Nextel Braid'    'Nextel Braid'   ;%   'XC' Type
                    'Poor'          'Good'        'Poor'          'Excellent'   'Good'       'Poor'        'Excellent'   'Fair'
'Silica'          'Silica'         ;%   'XS' Type
                    'Excellent'     'Good'        'Excellent'     'Excellent'   'Excellent'  'Excellent'   'Excellent'
'Excellent'   'TFE Teflon'      'TFE Teflon'     };%   'TFE' Type

%  Combine Arrays along the second dimension
TCWIns_DB = cat(2,TCWIns_DB1,TCWIns_DB2);

tmp = 1;
for t = 1:size(TEMPSENSOR_DB,1)
```

```
    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
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('\n\n  ***  WARNING  ***');
     fprintf('\n  NO VIABLE TEMPERATURE SENSOR EXISTS IN DATABASE!');
     fprintf('\n  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
     else
         SENS_NUM = c;
         break;
     end

    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);



%  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('\n\n   ***   T E M P E R A T U R E   S E N S O R   S U M M A R Y   R E S U L T S   ***');
 fprintf('\n\n***   I N P U T S   ***');
 fprintf('\n  Planetary Surface Temperature:   %7.2f  %s', PLTEMP, TempUnit);
```

```
        fprintf('\n  Unit System:          %s', UNITS);

        fprintf('\n\n*************************************************************************\n\n');

        %  Temperature Sensor Properties
        fprintf('\n\n** Sensor Properties **');
        fprintf('\n  Thermal Sensor Type:           %s', MATRIX2{SENS_NUM,1});
        fprintf('\n  Thermal Sensor Low Limit:   %9.3f   %s', MATRIX2{SENS_NUM,2}, TempUnitLabelStr{2} );
        fprintf('\n  Thermal Sensor High Limit:  %9.3f   %s', MATRIX2{SENS_NUM,3}, TempUnitLabelStr{3} );
        fprintf('\n  Power Requirements:         %9.3f   %s', MATRIX2{SENS_NUM,4}, TempUnitLabelStr{4} );
        fprintf('\n  Mass Requirements:          %9.3f   %s', MATRIX2{SENS_NUM,5}, TempUnitLabelStr{5} );
        fprintf('\n  Dimensions - Length:        %9.3f   %s', MATRIX2{SENS_NUM,6}, TempUnitLabelStr{6} );
        fprintf('\n               Width:         %9.3f   %s', MATRIX2{SENS_NUM,7}, TempUnitLabelStr{7} );
        fprintf('\n              Height:         %9.3f   %s', MATRIX2{SENS_NUM,8}, TempUnitLabelStr{8} );
        fprintf('\n  Operating Pressure:         %9.3f   %s', MATRIX2{SENS_NUM,9}, TempUnitLabelStr{9} );
        fprintf('\n  Error Tolerance:            %9.3f   %s', MATRIX2{SENS_NUM,10}, TempUnitLabelStr{10} );
        fprintf('\n                              %9.3f   %s', MATRIX2{SENS_NUM,11}, TempUnitLabelStr{11} );
        fprintf('\n  TRL Level:                  %9d     %s', MATRIX2{SENS_NUM,12}, TempUnitLabelStr{12} );

        fprintf('\n\nThermocouple Construction Materials');
        fprintf('\n  Positive Terminal:         %10s   %6s', MATRIX2{SENS_NUM,13}, TempUnitLabelStr{13} );
        fprintf('\n  Negative Terminal:         %10s   %6s', MATRIX2{SENS_NUM,14}, TempUnitLabelStr{14} );

        fprintf('\n\nEMF Voltage spread over full Temperature Range:  %7.3f  %s', MATRIX2{SENS_NUM,15}+MATRIX2{SENS_NUM,16},
TempUnitLabelStr{15});

        fprintf('\n\n  Sensor Comments:');
        fprintf('\n  %s', MATRIX2{SENS_NUM,19});


        fprintf('\n\n     INSULATION MATERIAL PROPERTIES');
        fprintf('\n  Insulation Type:              %s', INS_MATRIX{Max_INS,1});
        fprintf('\n  Insulation Lower Limit:     %9.3f   %s', INS_MATRIX{Max_INS,2},WInsUnitLabelStr{2});
        fprintf('\n  Insulation Upper Limit:     %9.3f   %s', INS_MATRIX{Max_INS,3},WInsUnitLabelStr{3});

        fprintf('\n\n  Environmental Properties');
        fprintf('\n  Abrasion Resistance:              %s', INS_MATRIX{Max_INS,4});
        fprintf('\n  Flexibility:                      %s', INS_MATRIX{Max_INS,5});
        fprintf('\n  Water Submersion:                 %s', INS_MATRIX{Max_INS,6});
        fprintf('\n  Resistance to Solvents:           %s', INS_MATRIX{Max_INS,7});
        fprintf('\n  Resistance to Acids:              %s', INS_MATRIX{Max_INS,8});
        fprintf('\n  Resistance to Bases:              %s', INS_MATRIX{Max_INS,9});
        fprintf('\n  Resistance to Flame:              %s', INS_MATRIX{Max_INS,10});
        fprintf('\n  Resistance to Humidity:           %s', INS_MATRIX{Max_INS,11});

        fprintf('\n  Overall Material Construction:    %s', INS_MATRIX{Max_INS,12});
        fprintf('\n  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   ***   T E M P E R A T U R E   S E N S O R   S U M M A R Y   R E S U L T S   ***');
fprintf(Fout, '\n\n***   I N P U T S   ***');
fprintf(Fout, '\n  Planetary Surface Temperature:   %7.2f  %s', PLTEMP, TempUnit);
fprintf(Fout, '\n  Unit System:          %s', UNITS);

fprintf(Fout, '\n\n*********************************************************************\n\n');

%  Temperature Sensor Properties
fprintf(Fout, '\n\n** Sensor Properties **');
fprintf(Fout, '\n  Thermal Sensor Type:    %s', MATRIX2{SENS_NUM,1});
fprintf(Fout, '\n  Thermal Sensor Low Limit:   %9.3f   %s', MATRIX2{SENS_NUM,2}, TempUnitLabelStr{2} );
fprintf(Fout, '\n  Thermal Sensor High Limit:  %9.3f   %s', MATRIX2{SENS_NUM,3}, TempUnitLabelStr{3} );
fprintf(Fout, '\n  Power Requirements:         %9.3f   %s', MATRIX2{SENS_NUM,4}, TempUnitLabelStr{4} );
fprintf(Fout, '\n  Mass Requirements:          %9.3f   %s', MATRIX2{SENS_NUM,5}, TempUnitLabelStr{5} );
fprintf(Fout, '\n  Dimensions - Length:        %9.3f   %s', MATRIX2{SENS_NUM,6}, TempUnitLabelStr{6} );
fprintf(Fout, '\n               Width:         %9.3f   %s', MATRIX2{SENS_NUM,7}, TempUnitLabelStr{7} );
fprintf(Fout, '\n              Height:         %9.3f   %s', MATRIX2{SENS_NUM,8}, TempUnitLabelStr{8} );
fprintf(Fout, '\n  Operating Pressure:         %9.3f   %s', MATRIX2{SENS_NUM,9}, TempUnitLabelStr{9} );
fprintf(Fout, '\n  Error Tolerance:            %9.3f   %s', MATRIX2{SENS_NUM,10}, TempUnitLabelStr{10} );
fprintf(Fout, '\n                              %9.3f   %s', MATRIX2{SENS_NUM,11}, TempUnitLabelStr{11} );
fprintf(Fout, '\n  TRL Level:                  %9d     %s', MATRIX2{SENS_NUM,12}, TempUnitLabelStr{12} );

fprintf(Fout, '\n\nThermocouple Construction Materials');
fprintf(Fout, '\n  Positive Terminal:        %10s   %6s', MATRIX2{SENS_NUM,13}, TempUnitLabelStr{13} );
fprintf(Fout, '\n  Negative Terminal:        %10s   %6s', MATRIX2{SENS_NUM,14}, TempUnitLabelStr{14} );

fprintf(Fout, '\n\nEMF Voltage spread over full Temperature Range:  %7.3f  %s', MATRIX2{SENS_NUM,15}+MATRIX2{SENS_NUM,16},
TempUnitLabelStr{15});

fprintf(Fout, '\n\n  Sensor Comments:');
fprintf(Fout, '\n  %s', MATRIX2{SENS_NUM,19});

fprintf(Fout, '\n\n     INSULATION MATERIAL PROPERTIES');
fprintf(Fout, '\n  Insulation Type:                %s', INS_MATRIX{Max_INS,1});
fprintf(Fout, '\n  Insulation Lower Limit:        %9.3f   %s', INS_MATRIX{Max_INS,2},WInsUnitLabelStr{2});
fprintf(Fout, '\n  Insulation Upper Limit:        %9.3f   %s', INS_MATRIX{Max_INS,3},WInsUnitLabelStr{3});

fprintf(Fout, '\n\n  Environmental Properties');
fprintf(Fout, '\n  Abrasion Resistance:                %s', INS_MATRIX{Max_INS,4});
fprintf(Fout, '\n  Flexibility:                        %s', INS_MATRIX{Max_INS,5});
fprintf(Fout, '\n  Water Submersion:                   %s', INS_MATRIX{Max_INS,6});
fprintf(Fout, '\n  Resistance to Solvents:             %s', INS_MATRIX{Max_INS,7});
fprintf(Fout, '\n  Resistance to Acids:                %s', INS_MATRIX{Max_INS,8});
fprintf(Fout, '\n  Resistance to Bases:                %s', INS_MATRIX{Max_INS,9});
fprintf(Fout, '\n  Resistance to Flame:                %s', INS_MATRIX{Max_INS,10});
fprintf(Fout, '\n  Resistance to Humidity:             %s', INS_MATRIX{Max_INS,11});

fprintf(Fout, '\n  Overall Material Construction:      %s', INS_MATRIX{Max_INS,12});
fprintf(Fout, '\n  Conductor Material Construction:    %s', INS_MATRIX{Max_INS,13});

fclose(Fout);
```

```
end


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('\n\n  ***  WARNING  ***');
          fprintf('\n  NO VIABLE TEMPERATURE SENSOR EXISTS IN DATABASE!');
          fprintf('\n  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

          %  Copies MATRIX2 to MATRIX3 if all Sensors Eliminated
          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
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('\n\n   ***   T E M P E R A T U R E   S E N S O R   S U M M A R Y   R E S U L T S   ***');
        fprintf('\n\n***   I N P U T S   ***');
        fprintf('\n  Planetary Surface Temperature:   %7.2f  %s', PLTEMP, TempUnit);
        fprintf('\n  Unit System:           %s', UNITS);

        fprintf('\n\n*********************************************************************\n\n');

        %  Temperature Sensor Properties
        fprintf('\n\n** Sensor Properties **');
        fprintf('\n  Thermal Sensor Type:                  %s', MATRIX2{1,1});
        fprintf('\n  Thermal Sensor Low Limit:   %9.3f   %s', MATRIX2{1,2}, TempUnitLabelStr{2} );
        fprintf('\n  Thermal Sensor High Limit:  %9.3f   %s', MATRIX2{1,3}, TempUnitLabelStr{3} );
        fprintf('\n  Power Requirements:         %9.3f   %s', MATRIX2{1,4}, TempUnitLabelStr{4} );
        fprintf('\n  Mass Requirements:          %9.3f   %s', MATRIX2{1,5}, TempUnitLabelStr{5} );
        fprintf('\n  Dimensions - Length:        %9.3f   %s', MATRIX2{1,6}, TempUnitLabelStr{6} );
        fprintf('\n               Width:         %9.3f   %s', MATRIX2{1,7}, TempUnitLabelStr{7} );
        fprintf('\n              Height:         %9.3f   %s', MATRIX2{1,8}, TempUnitLabelStr{8} );
        fprintf('\n  Operating Pressure:         %9.3f   %s', MATRIX2{1,9}, TempUnitLabelStr{9} );
        fprintf('\n  Error Tolerance:            %9.3f   %s', MATRIX2{1,10}, TempUnitLabelStr{10} );
        fprintf('\n                              %9.3f   %s', MATRIX2{1,11}, TempUnitLabelStr{11} );
        fprintf('\n  TRL Level:                  %9d     %s', MATRIX2{1,12}, TempUnitLabelStr{12} );

        fprintf('\n\nThermocouple Construction Materials');
        fprintf('\n  Positive Terminal:         %10s   %6s', MATRIX2{1,13}, TempUnitLabelStr{13} );
        fprintf('\n  Negative Terminal:         %10s   %6s', MATRIX2{1,14}, TempUnitLabelStr{14} );

        fprintf('\n\nEMF Voltage spread over full Temperature Range:  %7.3f  %s', MATRIX2{1,15}+MATRIX2{1,16},
TempUnitLabelStr{15});

        fprintf('\n\n  Sensor Comments:');
        fprintf('\n  %s', MATRIX2{1,19});

        end


        %*************************************************************************
        %***     Writes a text file summary of the Planetary Properties

    Fout = fopen('TEMPERATURE_SENSOR_Summary.txt','w+');

    fprintf(Fout, '\n\n   ***   T E M P E R A T U R E   S E N S O R   S U M M A R Y   R E S U L T S   ***');
    fprintf(Fout, '\n\n***   I N P U T S   ***');
```

```
    fprintf(Fout, '\n  Planetary Surface Temperature:   %7.2f  %s', PLTEMP, TempUnit);
    fprintf(Fout, '\n  Unit System:           %s', UNITS);

    fprintf(Fout, '\n\n*********************************************************************\n\n');

    %  Temperature Sensor Properties
    fprintf(Fout, '\n\n** Sensor Properties **');
    fprintf(Fout, '\n  Thermal Sensor Type:    %s', MATRIX2{1,1});
    fprintf(Fout, '\n  Thermal Sensor Low Limit:   %9.3f   %s', MATRIX2{1,2}, TempUnitLabelStr{2} );
    fprintf(Fout, '\n  Thermal Sensor High Limit:  %9.3f   %s', MATRIX2{1,3}, TempUnitLabelStr{3} );
    fprintf(Fout, '\n  Power Requirements:         %9.3f   %s', MATRIX2{1,4}, TempUnitLabelStr{4} );
    fprintf(Fout, '\n  Mass Requirements:          %9.3f   %s', MATRIX2{1,5}, TempUnitLabelStr{5} );
    fprintf(Fout, '\n  Dimensions - Length:        %9.3f   %s', MATRIX2{1,6}, TempUnitLabelStr{6} );
    fprintf(Fout, '\n               Width:         %9.3f   %s', MATRIX2{1,7}, TempUnitLabelStr{7} );
    fprintf(Fout, '\n              Height:         %9.3f   %s', MATRIX2{1,8}, TempUnitLabelStr{8} );
    fprintf(Fout, '\n  Operating Pressure:         %9.3f   %s', MATRIX2{1,9}, TempUnitLabelStr{9} );
    fprintf(Fout, '\n  Error Tolerance:            %9.3f   %s', MATRIX2{1,10}, TempUnitLabelStr{10} );
    fprintf(Fout, '\n                              %9.3f   %s', MATRIX2{1,11}, TempUnitLabelStr{11} );
    fprintf(Fout, '\n  TRL Level:                  %9d     %s', MATRIX2{1,12}, TempUnitLabelStr{12} );

    fprintf(Fout, '\n\nThermocouple Construction Materials');
    fprintf(Fout, '\n  Positive Terminal:         %10s   %6s', MATRIX2{1,13}, TempUnitLabelStr{13} );
    fprintf(Fout, '\n  Negative Terminal:         %10s   %6s', MATRIX2{1,14}, TempUnitLabelStr{14} );

    fprintf(Fout, '\n\nEMF Voltage spread over full Temperature Range:  %7.3f  %s', MATRIX2{1,15}+MATRIX2{1,16},
TempUnitLabelStr{15});

    fprintf(Fout, '\n\n  Sensor Comments:');
    fprintf(Fout, '\n  %s', MATRIX2{1,19});

    fclose(Fout);

end
```

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              [       SENSOR       WIND VELOCITY     ACCURACY    DISP VOL FLOW RATE     PROBE TEMP RANGE
INST TEMP RANGE       POWER REQUIREMENT    ]
     %    Units                 [        TYPE        Low   m/s  High       % FS           N/A               Low   deg C  High
Low   deg C  High    Number       TYPE     ]
        WIND_VELOCITY_SENSOR_DB1 = {  '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      2          'AA'      ;
                                      'HHF42'     0.0000     20.015     1.00          'NO'               0.0           50.0
0.0         50.0      6          'AAA'     ;
                                     'HHF710'     0.2032     39.624     0.50          'NO'             -30.0          100.0
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'     0.6350     24.892     2.00          'YES'            -10.0           50.0
-10.0         50.0      1       '9 Volt'    ;
                                     'HHF92A'     0.4000     35.000     3.00          'YES'            -10.0           50.0
-10.0         50.0      1       '9 Volt'    ;
                                'REED YK-80AS'    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             [       WEIGHT      DIMENSIONS HANDHELD - mm      DIMENSIONS PROBE - mm       VOLTAGE
TYPE      CURRENT    POWER     ]
        %    Units                [         kg         Length    Width    Height     Length    Width    Height        V
N/A         mA        mW       ]
```

```
        WIND_VELOCITY_SENSOR_DB2 = {     0.227       180.34    76.20     20.32      124.0     70.0      41.0        0
'N/A'       0          0     ;
                                         0.227       180.34    76.20     20.32      180.3     25.4      25.4        0
'N/A'       0          0     ;
                                         0.355       185.00    78.00     38.00      280.0     12.0      12.0        0
'N/A'       0          0     ;
                                           0         150.00    81.00     30.00      178.0     70.0      70.0        0
'N/A'       0          0     ;
                                         0.227       180.00    76.00     20.00      180.0     70.0      70.0        0
'N/A'       0          0     ;
                                         0.227       180.00    76.00     20.00      180.0     70.0      70.0        0
'N/A'       0          0     ;
                                         1.400       181.00    70.00     35.00      181.0     73.0      73.0        0
'N/A'       0          0     ;
                                           0         183.00    76.00     45.00        0        0         0         0
'N/A'       0          0     ;
                                         0.220       200.00    68.00     30.00      600.0     13.0      13.0        0
'N/A'       0          0     ;
                                         0.210       181.00    76.00     45.00        0        0         0         0
'N/A'       0          0     ;
                                         0.500         0         0         0        162.0     50.0      50.0        7.0
'DC'      14.0       100.0  };

       WVelocityUnitLabelStr1 = { 'N/A'  'm/s'  'm/s'   '% FS'  'N/A'  'deg C'  'deg C'   'deg C'  'deg C'  'N/A'  'Type'  };
       WVelocityUnitLabelStr2 = {  'kg'   'mm'  'mm'   'mm'   'mm'   'mm'   'mm'    'V'    'N/A'    'mA'    'mW'   };

       end

       if strcmpi(UNITS,'British') == 1
    %  Properties              [       SENSOR      WIND VELOCITY     ACCURACY    DISP VOL FLOW RATE     PROBE TEMP RANGE
INST TEMP RANGE       POWER REQUIREMENT    ]
    %    Units                 [        TYPE      Low  fpm   High     % FS              N/A              Low   deg F  High
Low   deg F  High     Number     TYPE     ]
       WIND_VELOCITY_SENSOR_DB1 = {  'HH-30A'    40.00      7800.0     0.50         'NO'              -4.0          212.0
32.0        122.0      2          'AA'      ;
                                        'HHF300A'    60.00      6800.0     0.50         'NO'              -4.0          212.0
32.0        122.0      2          'AA'      ;
                                         'HHF42'     0.00      3490.0     1.00         'NO'              32.0          122.0
32.0        122.0      6          'AAA'     ;
                                        'HHF710'     40.00      7800.0     0.50         'NO'              -4.0          212.0
32.0        212.0      0      'NiCad w/AC'  ;
                                        'HHF750'     40.00      7800.0     0.50         'YES'             -4.0          212.0
32.0        122.0      2          'AA'      ;
                                        'HHF753'     40.00      7800.0     0.50         'NO'              -4.0          176.0
32.0        122.0      2          'AA'      ;
                                         'HHF91'    125.00      4900.0     2.00         'YES'             14.0          122.0
-10.0        122.0      1        '9 Volt'    ;
                                        'HHF92A'     80.00      5900.0     3.00         'YES'             14.0          122.0
-10.0        122.0      1        '9 Volt'    ;
                                       'REED YK-80AS'  160.00     2358.0     2.00         'NO'              32.0          122.0
32.0        122.0      1        '9 Volt'    ;
                                         'REED 8903'     80.00      5900.0       0          'NO'              14.0          122.0
14.0        122.0      1        '9 Volt'    ;
```

```
                                        'FT 702'       0.00     13779.5     4.00         'NO'              -40.0         185.0
-40.0        185.0       0          'N/A'     };

     %  Properties              [      WEIGHT      DIMENSIONS HANDHELD - in      DIMENSIONS PROBE - in         VOLTAGE     TYPE
CURRENT     POWER     ]
     %    Units                 [       lb         Length     Width     Height     Length     Width     Height          V       N/A
mA       BTUs/hr   ]
       WIND_VELOCITY_SENSOR_DB2 = {   0.500          7.10      3.00       0.80       4.875     2.750     1.635          0       'N/A'
0           0      ;                  0.500          7.10      3.00       0.80       7.100     1.000     1.000          0       'N/A'
0           0      ;                  0.771          7.10      3.10       1.50      11.000     0.500     0.500          0       'N/A'
0           0      ;                    0            5.90      3.20       1.20       7.000     2.750     2.750          0       'N/A'
0           0      ;                  0.500          7.10      3.00       0.80       7.100     2.750     2.750          0       'N/A'
0           0      ;                  0.500          7.10      3.00       0.80       7.100     2.750     2.750          0       'N/A'
0           0      ;                  3.086          7.12      2.75       1.37       7.120     2.875     2.875          0       'N/A'
0           0      ;                    0            7.20      3.00       1.80         0         0         0            0       'N/A'
0           0      ;                  0.480          7.90      2.70       1.20      23.622     0.512     0.512          0       'N/A'
0           0      ;                  0.463          7.10      2.75       1.40         0         0         0            0       'N/A'
0           0      ;                  1.102          6.38      1.97       1.97         0         0         0          7.0       'DC'
14.0        0.342   };

        WVelocityUnitLabelStr1 = { 'N/A'  'ft/min'  'ft/min'  '% FS'  'N/A'  'deg F'  'deg F'   'deg F'  'deg F'  'N/A'  'Type'
};
        WVelocityUnitLabelStr2 = {   'lb'   'in'  'in'  'in'  'in'  'in'  'in'   'V'   'N/A'   'mA'   'BTU/hr'   };

        end

        WIND_VELOCITY_SENSOR_DB3{ 1,1} = 'Compact Design. 3 Extension Rods. 2 or 16 sec Averaging Period. 1"/2.75" Probe
Diameters.';
        WIND_VELOCITY_SENSOR_DB3{ 2,1} = 'Compact Design. 2 Extension Rods. 2-sec averaging period. 1" Probe Diameter.';
        WIND_VELOCITY_SENSOR_DB3{ 3,1} = 'Combination Velocity/Temperature Probe. Min/Max Velocity Measurements. 0.5" Diameter
Telescoping Probe.  Low Flow Speeds.';
        WIND_VELOCITY_SENSOR_DB3{ 4,1} = 'Combination Velocity/Temperature/Humidity Probe. Min/Max/Avg Velocity Measurements.
1"/2.75" Probe Diameters. RS232/Analog Signal Output Format.';
        WIND_VELOCITY_SENSOR_DB3{ 5,1} = 'Combination Velocity/Temperature Probe. Min/Max/Avg Velocity Measurements.  1"/2.75"
Probe Diameters.  2 or 16 sec Averaging Period.';
        WIND_VELOCITY_SENSOR_DB3{ 6,1} = 'Combination Velocity/Temperature/Humidity Probe. Min/Max/Avg Velocity Measurements.
1"/2.75" Probe Diameters.';
        WIND_VELOCITY_SENSOR_DB3{ 7,1} = 'Volumetric Flow Rate Capability.  Temperature Measurement.  Min/Max Velocity
Measurements.  RS232 Signal Output Format.';
        WIND_VELOCITY_SENSOR_DB3{ 8,1} = 'Volumetric Flow Rate Capability.  Temperature Measurement.  Min/Max/Avg Velocity
Measurements. Integrated Probe/Electronics configuration.';
        WIND_VELOCITY_SENSOR_DB3{ 9,1} = 'Combination Velocity/Temperature Probe. Min/Max Velocity Measurements. Telescoping
Wand Probe. RS232 Signal Output Format.';
```

```
        WIND_VELOCITY_SENSOR_DB3{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_DB3{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,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_MEProp{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 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                                                  Instrument
Dimensions
```

```
                WV_SCALE_FAC(mv) = W_MATRIX{mv,12} * ( (W_MATRIX{mv,13} * W_MATRIX{mv,14} * W_MATRIX{mv,15}) + (W_MATRIX{mv,16}
* 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_INDX ] = min(W_MATRIX2);

                W_MATRIX3 = W_MATRIX2{ACC_INDX,:};
            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

        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
  W_MATRIX3{1,1} = 'N/A';
  W_MATRIX3{1,5} = 'N/A';
  W_MATRIX3{1,11} = 'N/A';
  W_MATRIX3{1,19} = '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('\n\n   ***   W I N D   V E L O C I T Y   S E N S O R   S U M M A R Y   R E S U L T S   ***');
 fprintf('\n\n***   I N P U T S   ***');
 fprintf('\n  Wind Velocity Sensor Type:     %s', WV_TYPE);
 fprintf('\n  Unit System:           %s', UNITS);

 fprintf('\n\n*************************************************************************\n');

 %  Wind Velocity Sensor Properties
 fprintf('\n\n** Sensor Properties **');
 fprintf('\n  Wind Velocity Sensor Type:          %s', W_MATRIX3{1,1});
 fprintf('\n  Velocity Sensing Range - Low:   %9.4f   %s', W_MATRIX3{1,2}, WVelocityUnitLabelStr{2} );
 fprintf('\n                          High:   %9.4f   %s', W_MATRIX3{1,3}, WVelocityUnitLabelStr{3} );
 fprintf('\n  Velocity Accuracy:              %9.4f   %s', W_MATRIX3{1,4}, WVelocityUnitLabelStr{4} );
 fprintf('\n  Volumetric Flow Rate:           %s', W_MATRIX3{1,5} );

 %  Environmental Properties
 fprintf('\n\n   Environmental Properties');
 fprintf('\n  Probe Temperature Range - Low:        %9.3f   %s', W_MATRIX3{1,6}, WVelocityUnitLabelStr{6} );
 fprintf('\n                            High:        %9.3f   %s', W_MATRIX3{1,7}, WVelocityUnitLabelStr{7} );
 fprintf('\n  Instrument Temperature Range - Low:   %9.3f   %s', W_MATRIX3{1,8}, WVelocityUnitLabelStr{8} );
 fprintf('\n                                High:   %9.3f   %s', W_MATRIX3{1,9}, WVelocityUnitLabelStr{9} );

 fprintf('\n\n  Power Requirements');
 fprintf('\n  Number of Batteries:           %9d   %s',  W_MATRIX3{1,10}, WVelocityUnitLabelStr{10} );
 fprintf('\n  Battery Type:                          %s   %s', W_MATRIX3{1,11}, WVelocityUnitLabelStr{11} );
 fprintf('\n  Voltage Input:                 %9.3f   %s', W_MATRIX3{1,19}, WVelocityUnitLabelStr{19} );
 fprintf('\n  Voltage Type:                          %s   %s', W_MATRIX3{1,20}, WVelocityUnitLabelStr{20} );
 fprintf('\n  Current Input:                 %9.3f   %s', W_MATRIX3{1,21}, WVelocityUnitLabelStr{21} );
 fprintf('\n  Input Power:                   %9.3f   %s', W_MATRIX3{1,22}, WVelocityUnitLabelStr{22} );

 fprintf('\n\n  Physical Properties');
 fprintf('\n  Sensor Mass:                   %9.3f   %s', W_MATRIX3{1,12}, WVelocityUnitLabelStr{12} );
 fprintf('\n  Handheld Dimensions - Length:  %9.3f   %s', W_MATRIX3{1,13}, WVelocityUnitLabelStr{13} );
 fprintf('\n                         Width:  %9.3f   %s', W_MATRIX3{1,14}, WVelocityUnitLabelStr{14} );
 fprintf('\n                        Height:  %9.3f   %s', W_MATRIX3{1,15}, WVelocityUnitLabelStr{15} );
 fprintf('\n  Probe Dimensions - Length:     %9.3f   %s', W_MATRIX3{1,16}, WVelocityUnitLabelStr{16} );
```

```
fprintf('\n                        Width:     %9.3f   %s', W_MATRIX3{1,17}, WVelocityUnitLabelStr{17} );
fprintf('\n                       Height:     %9.3f   %s', W_MATRIX3{1,18}, WVelocityUnitLabelStr{18} );

fprintf('\n\n  Sensor Comments:');
fprintf('\n  %s', W_MATRIX3{1,23});

end


%************************************************************************
%***     Writes a text file summary of the Planetary Properties

Fout = fopen('WIND_VELOCITY_SENSOR_Summary.txt','w+');

fprintf(Fout, '\n\n   ***   W I N D   V E L O C I T Y   S E N S O R   S U M M A R Y   R E S U L T S   ***');
fprintf(Fout, '\n\n***   I N P U T S   ***');
fprintf(Fout, '\n  Wind Velocity Sensor Type:     %s', WV_TYPE);
fprintf(Fout, '\n  Unit System:           %s', UNITS);

fprintf(Fout, '\n\n************************************************************************\n');

%  Wind Velocity Sensor Properties
fprintf(Fout, '\n\n** Sensor Properties **');
fprintf(Fout, '\n  Wind Velocity Sensor Type:          %s', W_MATRIX3{1,1});
fprintf(Fout, '\n  Velocity Sensing Range - Low:    %9.4f   %s', W_MATRIX3{1,2}, WVelocityUnitLabelStr{2} );
fprintf(Fout, '\n                          High:    %9.4f   %s', W_MATRIX3{1,3}, WVelocityUnitLabelStr{3} );
fprintf(Fout, '\n  Velocity Accuracy:               %9.4f   %s', W_MATRIX3{1,4}, WVelocityUnitLabelStr{4} );
fprintf(Fout, '\n  Volumetric Flow Rate:            %s', W_MATRIX3{1,5} );

%  Environmental Properties
fprintf(Fout, '\n\n   Environmental Properties');
fprintf(Fout, '\n  Probe Temperature Range - Low:        %9.3f   %s', W_MATRIX3{1,6}, WVelocityUnitLabelStr{6} );
fprintf(Fout, '\n                            High:       %9.3f   %s', W_MATRIX3{1,7}, WVelocityUnitLabelStr{7} );
fprintf(Fout, '\n  Instrument Temperature Range - Low:   %9.3f   %s', W_MATRIX3{1,8}, WVelocityUnitLabelStr{8} );
fprintf(Fout, '\n                                 High:   %9.3f   %s', W_MATRIX3{1,9}, WVelocityUnitLabelStr{9} );

fprintf(Fout, '\n\n  Power Requirements');
fprintf(Fout, '\n  Number of Batteries:           %9d   %s',  W_MATRIX3{1,10}, WVelocityUnitLabelStr{10} );
fprintf(Fout, '\n  Battery Type:                          %s   %s', W_MATRIX3{1,11}, WVelocityUnitLabelStr{11} );
fprintf(Fout, '\n  Voltage Input:                 %9.3f   %s', W_MATRIX3{1,19}, WVelocityUnitLabelStr{19} );
fprintf(Fout, '\n  Voltage Type:                          %s   %s', W_MATRIX3{1,20}, WVelocityUnitLabelStr{20} );
fprintf(Fout, '\n  Current Input:                 %9.3f   %s', W_MATRIX3{1,21}, WVelocityUnitLabelStr{21} );
fprintf(Fout, '\n  Input Power:                   %9.3f   %s', W_MATRIX3{1,22}, WVelocityUnitLabelStr{22} );

fprintf(Fout, '\n\n  Physical Properties');
fprintf(Fout, '\n  Sensor Mass:                   %9.3f   %s', W_MATRIX3{1,12}, WVelocityUnitLabelStr{12} );
fprintf(Fout, '\n  Handheld Dimensions - Length:  %9.3f   %s', W_MATRIX3{1,13}, WVelocityUnitLabelStr{13} );
fprintf(Fout, '\n                          Width:  %9.3f   %s', W_MATRIX3{1,14}, WVelocityUnitLabelStr{14} );
fprintf(Fout, '\n                         Height:  %9.3f   %s', W_MATRIX3{1,15}, WVelocityUnitLabelStr{15} );
fprintf(Fout, '\n  Probe Dimensions - Length:     %9.3f   %s', W_MATRIX3{1,16}, WVelocityUnitLabelStr{16} );
fprintf(Fout, '\n                       Width:    %9.3f   %s', W_MATRIX3{1,17}, WVelocityUnitLabelStr{17} );
fprintf(Fout, '\n                      Height:    %9.3f   %s', W_MATRIX3{1,18}, WVelocityUnitLabelStr{18} );

fprintf(Fout, '\n\n  Sensor Comments:');
```

```
        fprintf(Fout, '\n  %s', W_MATRIX3{1,23});

        fclose(Fout);

end

%   Begins Selection Section For Doppler Type Wind Velocity Sensors
if strcmpi(WV_TYPE,'DOPPLER') == 1
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %%   Wind Velocity Sensor Data Base Created Based on un Unit System
        %%  2 - D Matrix  WIND_VELOCITY_SENSOR_DB(WV_Type, Property)


     %                          [  Electrical Specifications
     %  Properties              [        SENSOR   OUTPUT FREQUENCY   OUTPUT LEVEL    SPECTRAL PURITY      SPECTRAL PURITY
AGING DATA         ACCURACY     STABILITY   WARM UP TIME    VOLTAGE TYPE    VOLTAGE      ]
     %    Units                 [         TYPE          Mhz          dB   +- TOL        Harmonics          Non-Harmonics
Monthly  10 Years      @ 25 C     @ 100 sec       min              N/A         Volts       ]
        WIND_VELOCITY_SENSOR_DB1 =  {  '8130A'        10.0          7.0     1.50         -30.0              -80.0
5.0E-11   1.0E-09      1.0E-11     3.0E-12       14.0              'DC'      [ 22  32 ]    ;
                                    '9633'        10.0          7.0     2.00         -35.0              -80.0
3.3E-09   1.0E-06       0.00       5.0E-12        0.0              'DC'        [  15  ]    };

        WVelocityUnitLabelStr1 = { 'N/A'  'MHz'  'dBm'   'dBm'  'dBc'  'dBc'  'Hz/month '   '10 yrs'  'Hz @ 25 deg C'  'Hz @
100 sec'  'min'    'Type'    'V'     };

    if strcmpi(UNITS,'SI') == 1
     %  Properties             [        OPERATING TEMP RANGE      STORAGE TEMP RANGE    TEMP SENSITIVITY   ORIENTATION SENS
PRESSURE SENS      WARM-UP MAX POWER   STEADY STATE POWER   INPUT POWER QUIESCENT    WEIGHT          DIMENSIONS - cm
LIFETIME      ]
     %    Units                [         Low   deg C  High        Low   deg C  High       Op Temp Range          Hz
Hz/mbar            W @28 V            W @ 28 V          W @ 28 V @ 25 deg C      Kg          Length   Width    Height
years         ]
        WIND_VELOCITY_SENSOR_DB2 =  {     -40.0        68.0       -62.0        85.0        3.0E-10             5.0E-11
1.0E-13            35.0                22.0                12.0              0.900        10.26    7.41     7.28        20
;
                                          -40.0        70.0       -55.0        85.0        1.0E-08              0.00
0.00               12.0                4.0                 3.0               0.159         7.62    4.24     4.06         0
};

        WVelocityUnitLabelStr2 = { 'deg C'  'deg C'   'deg C'  'deg C'  'Hz'  'Hz'   'Hz/mbar'  'W'  'W'  'W'  'Kg'   'cm'
'cm'  'cm'  'years'   };
       end


       if strcmpi(UNITS,'British') == 1
     %  Properties             [        OPERATING TEMP RANGE      STORAGE TEMP RANGE    TEMP SENSITIVITY   ORIENTATION SENS
PRESSURE SENS      WARM-UP MAX POWER   STEADY STATE POWER     INPUT POWER QUIESCENT    WEIGHT          DIMENSIONS - in
LIFETIME      ]
     %    Units                [         Low   deg F  High        Low   deg F  High       Op Temp Range          Hz
Hz/psi            BTUs/hr @28 V       BTUs/hr @ 28 V    BTUs/hr @ 28 V @ 77 deg F    lbm         Length   Width    Height
years         ]
```

```
        WIND_VELOCITY_SENSOR_DB2 =  {     -40.0       154.4        -79.6       185.0         3.0E-10             5.0E-11
6.89E-12              119.351              75.134                  40.982              1.984        4.04      2.92      2.87
20         ;
                                          -40.0       158.0        -67.0       185.0         1.0E-08             0.00
0.00                  40.982              13.661                  10.245              0.351        3.00      1.67      1.60
0         };

        WVelocityUnitLabelStr2 = { 'deg F'  'deg F'   'deg F'  'deg F'  'Hz'  'Hz'   'Hz/psi'  'BTUs/hr'  'BTUs/hr'  'BTUs/hr
@77F'  'lbm'   'in'  'in'  'in'  'years'   };
        end

        WIND_VELOCITY_SENSOR_DB3{1,1} = {  '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'   };
    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_MEProp{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_VELOCITY_SENSOR_DB,1)
        W_DELTA(w) = WIND_VELOCITY_SENSOR_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)
        WV_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{mv,20} == 0)
            WV_SCALE_FAC(sv) = 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_MATRIX2,2)
                        W_MATRIX3{k,j} = W_MATRIX2{c,j};
                    end
                    k = k + 1;
                end
            end
        end

        %  Copies W_MATRIX2 to W_MATRIX3 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

    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
     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('\n\n   ***   W I N D   V E L O C I T Y   S E N S O R   S U M M A R Y   R E S U L T S   ***');
    fprintf('\n\n***   I N P U T S   ***');
```

```
fprintf('\n  Wind Velocity Sensor Type:     %s', WV_TYPE);
fprintf('\n  Unit System:          %s', UNITS);

fprintf('\n\n*************************************************************************\n');

%  Wind Velocity Sensor Properties
fprintf('\n\n** Sensor Properties **');
fprintf('\n  Wind Velocity Sensor Type:          %s', W_MATRIX3{1,1});
fprintf('\n  Sensor Output Frequency:          %9.3f   %s', W_MATRIX3{1,2}, WVelocityUnitLabelStr{2} );
fprintf('\n  Sensor Output Level:              %9.3f   %s', W_MATRIX3{1,3}, WVelocityUnitLabelStr{3} );
fprintf('\n  Output Tolerance:                 %9.3f   %s', W_MATRIX3{1,4}, WVelocityUnitLabelStr{4} );
fprintf('\n  Spectral Purity - Harmonics:      %9.3f   %s', W_MATRIX3{1,5}, WVelocityUnitLabelStr{5} );
fprintf('\n  Spectral Purity - Non-Harmonics:  %9.3f   %s', W_MATRIX3{1,6}, WVelocityUnitLabelStr{6} );
fprintf('\n  Aging Data - Month:               %9.2E   %s', W_MATRIX3{1,7}, WVelocityUnitLabelStr{7} );
fprintf('\n             - 10 years:            %9.2E   %s', W_MATRIX3{1,8}, WVelocityUnitLabelStr{8} );
fprintf('\n  Sensor Accuracy:                  %9.2E   %s', W_MATRIX3{1,9}, WVelocityUnitLabelStr{9} );
fprintf('\n  Sensor Stability:                 %9.2E   %s', W_MATRIX3{1,10}, WVelocityUnitLabelStr{10} );
fprintf('\n  Sensor Warm Up Time:              %9.3f   %s', W_MATRIX3{1,11}, WVelocityUnitLabelStr{11} );
fprintf('\n  Sensor Lifetime:                  %9.1f   %s', W_MATRIX3{1,28}, WVelocityUnitLabelStr{28} );

%  Environmental Properties
fprintf('\n\n   Environmental Properties');
fprintf('\n  Operating Temperature Range - Low:     %9.3f   %s', W_MATRIX3{1,14}, WVelocityUnitLabelStr{14} );
fprintf('\n                               High:     %9.3f   %s', W_MATRIX3{1,15}, WVelocityUnitLabelStr{15} );
fprintf('\n  Storage Temperature Range - Low:       %9.3f   %s', W_MATRIX3{1,16}, WVelocityUnitLabelStr{16} );
fprintf('\n                             High:       %9.3f   %s', W_MATRIX3{1,17}, WVelocityUnitLabelStr{17} );
fprintf('\n  Temperature Sensitivity - Op Range:    %9.2E   %s', W_MATRIX3{1,18}, WVelocityUnitLabelStr{18} );
fprintf('\n  Orientation Sensitivity:               %9.2E   %s', W_MATRIX3{1,19}, WVelocityUnitLabelStr{19} );
fprintf('\n  Pressure Sensitivity:                  %9.2E   %s', W_MATRIX3{1,20}, WVelocityUnitLabelStr{20} );

fprintf('\n\n  Power Requirements');
fprintf('\n  Voltage Type:                          %s   %s', W_MATRIX3{1,12}, WVelocityUnitLabelStr{12} );
fprintf('\n  System Voltage Levels:           ');
for g = 1:length(W_MATRIX3{1,13})
    fprintf('    %6.3f', W_MATRIX3{1,13}(g) );
end
fprintf('    %s', WVelocityUnitLabelStr{13} );
fprintf('\n  Warm Up Time Max Power:           %9.3f   %s', W_MATRIX3{1,21}, WVelocityUnitLabelStr{21} );
fprintf('\n  Steady State Power:               %9.3f   %s', W_MATRIX3{1,22}, WVelocityUnitLabelStr{22} );
fprintf('\n  Input Power - Quiescent:          %9.3f   %s', W_MATRIX3{1,23}, WVelocityUnitLabelStr{23} );

fprintf('\n\n  Physical Properties');
fprintf('\n  Sensor Mass:                    %9.3f   %s', W_MATRIX3{1,24}, WVelocityUnitLabelStr{24} );
fprintf('\n  Sensor Dimensions - Length:     %9.3f   %s', W_MATRIX3{1,25}, WVelocityUnitLabelStr{25} );
fprintf('\n                      Width:      %9.3f   %s', W_MATRIX3{1,26}, WVelocityUnitLabelStr{26} );
fprintf('\n                     Height:      %9.3f   %s', W_MATRIX3{1,27}, WVelocityUnitLabelStr{27} );

fprintf('\n\n  Sensor Comments:');
for r = 1:length(W_MATRIX3{1,28})
    fprintf('\n    %s', W_MATRIX3{1,29}{r} );
end

end
```

```
%************************************************************************
%***     Writes a text file summary of the Planetary Properties

Fout = fopen('WIND_VELOCITY_SENSOR_Summary.txt','w+');

fprintf(Fout, '\n\n   ***   W I N D   V E L O C I T Y   S E N S O R   S U M M A R Y   R E S U L T S   ***');
fprintf(Fout, '\n\n***   I N P U T S   ***');
fprintf(Fout, '\n  Wind Velocity Sensor Type:     %s', WV_TYPE);
fprintf(Fout, '\n  Unit System:           %s', UNITS);

fprintf(Fout, '\n\n************************************************************************\n');

 %  Wind Velocity Sensor Properties
fprintf(Fout, '\n\n** Sensor Properties **');
fprintf(Fout, '\n  Wind Velocity Sensor Type:          %s', W_MATRIX3{1,1});
fprintf(Fout, '\n  Sensor Output Frequency:          %9.3f   %s', W_MATRIX3{1,2}, WVelocityUnitLabelStr{2} );
fprintf(Fout, '\n  Sensor Output Level:              %9.3f   %s', W_MATRIX3{1,3}, WVelocityUnitLabelStr{3} );
fprintf(Fout, '\n  Output Tolerance:                 %9.3f   %s', W_MATRIX3{1,4}, WVelocityUnitLabelStr{4} );
fprintf(Fout, '\n  Spectral Purity - Harmonics:      %9.3f   %s', W_MATRIX3{1,5}, WVelocityUnitLabelStr{5} );
fprintf(Fout, '\n  Spectral Purity - Non-Harmonics:  %9.3f   %s', W_MATRIX3{1,6}, WVelocityUnitLabelStr{6} );
fprintf(Fout, '\n  Aging Data - Month:               %9.2E   %s', W_MATRIX3{1,7}, WVelocityUnitLabelStr{7} );
fprintf(Fout, '\n             - 10 years:            %9.2E   %s', W_MATRIX3{1,8}, WVelocityUnitLabelStr{8} );
fprintf(Fout, '\n  Sensor Accuracy:                  %9.2E   %s', W_MATRIX3{1,9}, WVelocityUnitLabelStr{9} );
fprintf(Fout, '\n  Sensor Stability:                 %9.2E   %s', W_MATRIX3{1,10}, WVelocityUnitLabelStr{10} );
fprintf(Fout, '\n  Sensor Warm Up Time:              %9.3f   %s', W_MATRIX3{1,11}, WVelocityUnitLabelStr{11} );
fprintf(Fout, '\n  Sensor Lifetime:                  %9.1f   %s', W_MATRIX3{1,28}, WVelocityUnitLabelStr{28} );

%  Environmental Properties
fprintf(Fout, '\n\n   Environmental Properties');
fprintf(Fout, '\n  Operating Temperature Range - Low:    %9.3f   %s', W_MATRIX3{1,14}, WVelocityUnitLabelStr{14} );
fprintf(Fout, '\n                              High:    %9.3f   %s', W_MATRIX3{1,15}, WVelocityUnitLabelStr{15} );
fprintf(Fout, '\n  Storage Temperature Range - Low:     %9.3f   %s', W_MATRIX3{1,16}, WVelocityUnitLabelStr{16} );
fprintf(Fout, '\n                             High:     %9.3f   %s', W_MATRIX3{1,17}, WVelocityUnitLabelStr{17} );
fprintf(Fout, '\n  Temperature Sensitivity - Op Range:  %9.2E   %s', W_MATRIX3{1,18}, WVelocityUnitLabelStr{18} );
fprintf(Fout, '\n  Orientation Sensitivity:             %9.2E   %s', W_MATRIX3{1,19}, WVelocityUnitLabelStr{19} );
fprintf(Fout, '\n  Pressure Sensitivity:                %9.2E   %s', W_MATRIX3{1,20}, WVelocityUnitLabelStr{20} );

fprintf(Fout, '\n\n  Power Requirements');
fprintf(Fout, '\n  Voltage Type:                          %s   %s', W_MATRIX3{1,12}, WVelocityUnitLabelStr{12} );
fprintf(Fout, '\n  System Voltage Levels:           ');
for g = 1:length(W_MATRIX3{1,13})
    fprintf(Fout, '    %6.3f', W_MATRIX3{1,13}(g) );
end
fprintf(Fout, '    %s', WVelocityUnitLabelStr{13} );
fprintf(Fout, '\n  Warm Up Time Max Power:          %9.3f   %s', W_MATRIX3{1,21}, WVelocityUnitLabelStr{21} );
fprintf(Fout, '\n  Steady State Power:              %9.3f   %s', W_MATRIX3{1,22}, WVelocityUnitLabelStr{22} );
fprintf(Fout, '\n  Input Power - Quiescent:         %9.3f   %s', W_MATRIX3{1,23}, WVelocityUnitLabelStr{23} );

fprintf(Fout, '\n\n  Physical Properties');
fprintf(Fout, '\n  Sensor Mass:                  %9.3f   %s', W_MATRIX3{1,24}, WVelocityUnitLabelStr{24} );
fprintf(Fout, '\n  Sensor Dimensions - Length:   %9.3f   %s', W_MATRIX3{1,25}, WVelocityUnitLabelStr{25} );
fprintf(Fout, '\n                      Width:    %9.3f   %s', W_MATRIX3{1,26}, WVelocityUnitLabelStr{26} );
fprintf(Fout, '\n                     Height:    %9.3f   %s', W_MATRIX3{1,27}, WVelocityUnitLabelStr{27} );
```

```
        fprintf(Fout, '\n\n  Sensor Comments:');
        for r = 1:length(W_MATRIX3{1,29})
            fprintf(Fout, '\n    %s', W_MATRIX3{1,29}{r} );
        end

        fclose(Fout);

end
```

D18. Refraction Sensors

```
%   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
%      http://www.fiso.com/index.php?module=CMS&id=26
%

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        REFRACTION INDEX RANGE   RESOLUTION    DRIFT      ACCURACY    WEIGHT    VOLTAGE   TYPE
CURRENT   POWER       DIMENSIONS -  cm        ]
```

```
 % Units              [      #            Low      N/A     High         RIU          RIU          %           g          V        N/A
mA       mW    Length    Width   Height    ]
    REFRACTION_DB1 = {   'TSPR2KXY-R'      1.333           1.500       5.0E-06       0          1.00         0          5       'DC'
100      500     3.00      0.70    1.50     ;
                          'RFI-NP'         1.000           1.700       1.0E-04       0          0.071        0          0       'N/A'
0          0    22.80      0.97    0.97     ;
                          'REF-Huy'        1.250           1.450       1.0E-03       0          1.00         0         15       'DC'
6.66      10    10.00      10.00   11.61    };

    % Properties       [    OPERATING TEMPERATURE    ]
    % Units            [    Low deg C  High deg C    ]
    REFRACTION_DB2 =  {     -10.0         70.0       ;
                              0.0        100.0       ;
                            -40.0         65.0       };

    RefUnitLabelStr1 = { 'sensor'  'N/A'  'N/A'  'RIU'  'RIU'  '%'  'g'  'V'  'Type'  'mA'  'mW'  'cm'  'cm'  'cm'  };
    RefUnitLabelStr2 = { 'deg C'  'deg C'   };


end

if strcmpi(UNITS,'BRITISH') == 1
 % Properties        [    Sensor        REFRACTION INDEX RANGE   RESOLUTION    DRIFT      ACCURACY    WEIGHT    VOLTAGE   TYPE
CURRENT     POWER         DIMENSIONS -  in        ]
 % Units              [      #            Low      N/A     High         RIU          RIU          %           oz         V        N/A
mA       BTU's/hr    Length    Width   Height    ]
    REFRACTION_DB1 = {   'TSPR2KXY-R'      1.333           1.50        5.0E-06       0          1.00         0          5       'DC'
100       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           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'  '%'  'oz'  'V'  'Type'  'mA'  'BTUs/hr'  'in'  'in'  'in'  };
    RefUnitLabelStr2 = { 'deg F'  'deg F'   };


end

    %comments on sensors
    REFRACTION_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'   };
    REFRACTION_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 = REFRACTION_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
```

```
            k = k + 1;
        end
        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);

    %  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

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
```

```
    fprintf('\n\n   ***   R E F R A C T I O N   S E N S O R   S U M M A R Y   R E S U L T S   ***');
    fprintf('\n\n***   I N P U T S   ***');
    fprintf('\n  Unit System:          %s', UNITS);

    fprintf('\n\n*************************************************************************\n');

    %  Refraction Sensor Properties
    fprintf('\n\n** Sensor Properties **');
    fprintf('\n  Refraction Sensor Type:                %s', REF_MATRIX3{1,1});
    fprintf('\n  Sensing Refraction Range - Min:    %12.5f   %s', REF_MATRIX3{1,2}, RefUnitLabelStr{2} );
    fprintf('\n                             Middle:    %12.5f   %s', REF_MATRIX3{1,3}, RefUnitLabelStr{3} );
    fprintf('\n  Sensing Resolution:                   %12.4E   %s', REF_MATRIX3{1,4}, RefUnitLabelStr{4} );
    fprintf('\n  Sensing Drift Rate:                   %12.5f   %s', REF_MATRIX3{1,5}, RefUnitLabelStr{5} );
    fprintf('\n  Sensor Accuracy:                      %12.5f   %s', REF_MATRIX3{1,6}, RefUnitLabelStr{6} );


    fprintf('\n\n  Power Requirements');
    fprintf('\n  Input Voltage:                        %12.3f   %s', REF_MATRIX3{1,8}, RefUnitLabelStr{8} );
    fprintf('\n  Voltage Type:                                  %s   %s', REF_MATRIX3{1,9}, RefUnitLabelStr{9} );
    fprintf('\n  Input Current:                        %12.3f   %s', REF_MATRIX3{1,10}, RefUnitLabelStr{10} );
    fprintf('\n  Input Power:                          %12.3f   %s', REF_MATRIX3{1,11}, RefUnitLabelStr{11} );


    fprintf('\n\n  Physical Properties');
    fprintf('\n  Sensor Mass:                  %12.2f   %s', REF_MATRIX3{1,7}, RefUnitLabelStr{7} );
    fprintf('\n  Sensor Dimensions - Length:   %12.2f   %s', REF_MATRIX3{1,12}, RefUnitLabelStr{12} );
    fprintf('\n                       Width:   %12.2f   %s', REF_MATRIX3{1,13}, RefUnitLabelStr{13} );
    fprintf('\n                      Height:   %12.2f   %s', REF_MATRIX3{1,14}, RefUnitLabelStr{14} );


    %  Environmental Properties
    fprintf('\n\n   Environmental Properties');
    fprintf('\n  Temperature Range - Low:       %12.2f   %s', REF_MATRIX3{1,15}, RefUnitLabelStr{15} );
    fprintf('\n                     High:       %12.2f   %s', REF_MATRIX3{1,16}, RefUnitLabelStr{16} );


    fprintf('\n\n  Sensor Comments:');
    for r = 1:length(REF_MATRIX3{1,17})
        fprintf('\n    %s', REF_MATRIX3{1,17}{r} );
    end

end


%************************************************************************
%***     Writes a text file summary of the Planetary Properties

    Fout = fopen('REFRACTION_SENSOR_Summary.txt','w+');

    fprintf(Fout, '\n\n   ***   R E F R A C T I O N   S E N S O R   S U M M A R Y   R E S U L T S   ***');
    fprintf(Fout, '\n\n***   I N P U T S   ***');
    fprintf(Fout, '\n  Unit System:          %s', UNITS);

    fprintf(Fout, '\n\n*************************************************************************\n');
```

```
%  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                             Middle:    %12.5f   %s', REF_MATRIX3{1,3}, RefUnitLabelStr{3} );
fprintf(Fout, '\n  Sensing Resolution:                %12.4E   %s', REF_MATRIX3{1,4}, RefUnitLabelStr{4} );
fprintf(Fout, '\n  Sensing Drift Rate:                %12.5f   %s', REF_MATRIX3{1,5}, RefUnitLabelStr{5} );
fprintf(Fout, '\n  Sensor Accuracy:                   %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  Voltage Type:                                %s   %s', REF_MATRIX3{1,9}, RefUnitLabelStr{9} );
fprintf(Fout, '\n  Input Current:                     %12.3f   %s', REF_MATRIX3{1,10}, RefUnitLabelStr{10} );
fprintf(Fout, '\n  Input Power:                       %12.3f   %s', REF_MATRIX3{1,11}, RefUnitLabelStr{11} );


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                       Width:    %12.2f   %s', REF_MATRIX3{1,13}, RefUnitLabelStr{13} );
fprintf(Fout, '\n                      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                     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, '\n    %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
%     http://www.ortodoxism.ro/datasheets/analogdevices/ADSP-2100KG.pdf
%     http://www.chipdocs.com/datasheets/datasheet-pdf/Analog-Devices/ADSP-2100.html
%

function DIGITAL_SIGNAL_PROCESSING(DIG_SIG_TYPE,UNITS,Print)


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


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%   Digital Signal Processing Sensor Data Base Created Based on un Unit System
%%  2 - D Matrix  DIGSIGPROC_DB(DSP_Type, Property)

if strcmpi(UNITS,'SI') == 1
 % Properties        [    Sensor         PROCESSOR SPEED    CPU CLOCK SPEED     CPU CORES      DATA MEMORY           PROGRAM MEMORY
COMPUTATIONAL UNITS    CYCLE TIME    EXTERNAL INTERRUPTS    ]
 % Units             [       #              Category             MHz                 #         Amount - K    Locs      Amount - K
Locs       Number of Units          ns               Number           ]
   DIGSIGPROC_DB1 = {  'ADSP-2100'            'LOW'              8.192               1         16            16        32            24
3                  80.0                4                ;
                    'ADSP-2115'          'MEDIUM'           25.000               1          0.5           1         1             1
3                  40.0                0                ;
                    'ADSP-2185'           'HIGH'            75.000               1         8             1         8             1
3                  13.3                6                };

    % Properties       [    OPERATING TEMPERATURE      STORAGE TEMPERATURE     CPU POWER REQ     WEIGHT          VOLTAGE
TYPE   CURRENT     CPU DIMENSIONS -  mm       ]
    % Units            [    Low deg C  High deg C     Low deg C  High deg C          mW              kg         Min   V   Max
N/A      mA      Length    Width   Height    ]
    DIGSIGPROC_DB2 =  {     -55.0        125.0        -65.0        150.0             790              0         -0.3      7.0
'DC'    100       33.83    33.83     9.12    ;
                    -55.0        125.0        -65.0        150.0             205              0         -0.3      7.0
'DC'     38       28.19    28.19     8.74    ;
                    -40.0         85.0        -65.0        150.0             110              0         -0.5      4.0
'DC'     38       16.20    16.20     1.60    };

    DSPUnitLabelStr1 = { 'sensor'  'N/A'  'MHz'  '#'  'K-Words'  '#'  'K-Words'  '#'  '#'  'ns'  '#'   };
    DSPUnitLabelStr2 = { 'deg C'  'deg C'  'deg C'  'deg C'   'mW'   'kg'   'V'   'V'   'N/A'   'mA'   'mm'   'mm'   'mm'   };


end
```

```
if strcmpi(UNITS,'BRITISH') == 1
 % Properties        [    Sensor         PROCESSOR SPEED   CPU CLOCK SPEED    CPU CORES      DATA MEMORY          PROGRAM MEMORY
COMPUTATIONAL UNITS    CYCLE TIME    EXTERNAL INTERRUPTS    ]
 % Units             [      #             Category            MHz                #         Amount - K   Locs     Amount - K
Locs       Number of Units          ns                Number           ]
   DIGSIGPROC_DB1 = {  'ADSP-2100'            'LOW'              8.192              1         16           16        32           24
3                  80.0                4                ;
                       'ADSP-2115'          'MEDIUM'           25.000             1          0.5          1         1            1
3                  40.0                0                ;
                       'ADSP-2185'          'HIGH'             75.000             1          8            1         8            1
3                  13.3                6                };

    % Properties       [    OPERATING TEMPERATURE      STORAGE TEMPERATURE     CPU POWER REQ    WEIGHT         VOLTAGE
TYPE   CURRENT     CPU DIMENSIONS - in        ]
    % Units            [    Low deg F  High deg F     Low deg F  High deg F       BTU'S/hr         lb        Min   V   Max
N/A      mA      Length    Width   Height    ]
    DIGSIGPROC_DB2 =  {     -67.0        257.0        -85.0        302.0           2.698           0        -0.3      7.0
'DC'     100       1.332    1.332    0.359    ;
                         -67.0        257.0        -85.0        302.0           0.700           0        -0.3      7.0
'DC'      38       1.110    1.110    0.344    ;
                         -40.0        185.0        -85.0        302.0           0.376           0        -0.5      4.0
'DC'      38       0.638    0.638    0.063    };

    DSPUnitLabelStr1 = { 'sensor'  'N/A'  'MHz'  '#'  'K-Words'  '#'  'K-Words'  '#'  '#'  'ns'  '#'   };
    DSPUnitLabelStr2 = { 'deg F'  'deg F'  'deg F'  'deg F'   'BTUs/hr'   'lbm'   'V'   'V'   'N/A'   'mA'   'in'   'in'   'in'
};


end

    %comments on sensors
    DIGSIGPROC_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'   };
    DIGSIGPROC_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'   ;
```

```
                                '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-Buffered 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'   };
    DIGSIGPROC_DB3{ 3,1} = { 'Single-Cycle Instruction Execution'   ;
                                'Single-Cycle Context Switch'   ;
                                '3-Bus Architecture Allows Dual Operand Fetches in Every Instruction Cycle'   ;
                                'Multifunction 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
```

```
            for j = 1:size(DSP_MATRIX,2)
                DSP_MATRIX2{k,j} = DSP_MATRIX{c,j};
            end
            k = k + 1;
        end
        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');


%*************************************************************************************************************************
%
%   ***   PRINT SUMMARY Output Section for 'Print' = 'Y'

if strcmpi(Print,'Y') == 1

    fprintf('\n\n   ***   D I G I T A L    S I G N A L    P R O C E S S I N G    S U M M A R Y    R E S U L T S   ***');
    fprintf('\n\n***   I N P U T S   ***');
    fprintf('\n  CPU Operating Speed:   %s', DIG_SIG_TYPE);
    fprintf('\n  Unit System:          %s', UNITS);

    fprintf('\n\n*************************************************************************\n');

    %  Digital Signal Processing Properties
    fprintf('\n\n** Sensor Properties **');
    fprintf('\n  Digital Signal Processing Unit Type:       %s', DSP_MATRIX2{1,1});
    fprintf('\n  CPU Operating Speed:                       %s', DSP_MATRIX2{1,2});
    fprintf('\n  CPU Clock Speed:                      %12.3f   %s', DSP_MATRIX2{1,3},DSPUnitLabelStr{3} );
    fprintf('\n  Number of CPU Processing Cores:       %12d   %s', DSP_MATRIX2{1,4},DSPUnitLabelStr{4} );
    fprintf('\n  Amount of On-Board Data Memory:       %12.1f   %s', DSP_MATRIX2{1,5},DSPUnitLabelStr{5} );
    fprintf('\n  Number of Data Bins:                  %12d   %s', DSP_MATRIX2{1,6},DSPUnitLabelStr{6} );
    fprintf('\n  Amount of On-Board Program Memory:    %12.1f   %s', DSP_MATRIX2{1,7},DSPUnitLabelStr{7} );
    fprintf('\n  Number of Memory Bins:                %12d   %s', DSP_MATRIX2{1,8},DSPUnitLabelStr{8} );
    fprintf('\n  Number of Computational Units:        %12d   %s', DSP_MATRIX2{1,9},DSPUnitLabelStr{9} );
    fprintf('\n  CPU Cycle Time:                       %12.3f   %s', DSP_MATRIX2{1,10},DSPUnitLabelStr{10} );
    fprintf('\n  Number of External Interrupts:        %12d   %s', DSP_MATRIX2{1,11},DSPUnitLabelStr{11} );


    fprintf('\n\n  Power Requirements');
    fprintf('\n  CPU Output Power:                     %12.3f   %s', DSP_MATRIX2{1,16},DSPUnitLabelStr{16} );
    fprintf('\n  Input Voltage - Min:                  %12.3f   %s', DSP_MATRIX2{1,18},DSPUnitLabelStr{18} );
    fprintf('\n                  Max:                  %12.3f   %s', DSP_MATRIX2{1,19},DSPUnitLabelStr{19} );
    fprintf('\n  Voltage Type:                                     %s   %s',  DSP_MATRIX2{1,20},DSPUnitLabelStr{20} );
    fprintf('\n  Input Current:                        %12.3f   %s', DSP_MATRIX2{1,21},DSPUnitLabelStr{21} );
```

```
    fprintf('\n\n  Physical Properties');
    fprintf('\n  Processor Core Mass:                    %12.2f   %s', DSP_MATRIX2{1,17},DSPUnitLabelStr{17} );
    fprintf('\n  Sensor Dimensions - Length:             %12.3f   %s', DSP_MATRIX2{1,22},DSPUnitLabelStr{22} );
    fprintf('\n                       Width:             %12.3f   %s', DSP_MATRIX2{1,23},DSPUnitLabelStr{23} );
    fprintf('\n                      Height:             %12.3f   %s', DSP_MATRIX2{1,24},DSPUnitLabelStr{24} );


    %  Environmental Properties
    fprintf('\n\n   Environmental Properties');
    fprintf('\n  Operating Temperature Range - Low:       %12.2f   %s', DSP_MATRIX2{1,12},DSPUnitLabelStr{12} );
    fprintf('\n                                High:       %12.2f   %s', DSP_MATRIX2{1,13},DSPUnitLabelStr{13} );
    fprintf('\n  Storage Temperature Range - Low:         %12.2f   %s', DSP_MATRIX2{1,14},DSPUnitLabelStr{14} );
    fprintf('\n                               High:       %12.2f   %s', DSP_MATRIX2{1,15},DSPUnitLabelStr{15} );

    fprintf('\n\n  Sensor Comments:');
    for r = 1:length(DSP_MATRIX2{1,25})
        fprintf('\n    %s', DSP_MATRIX2{1,25}{r} );
    end

end


%************************************************************************
%***     Writes a text file summary of the Planetary Properties

    Fout = fopen('DIGITAL_SIGNAL_PROCESSING_Summary.txt','w+');

    fprintf(Fout, '\n\n   ***   D I G I T A L    S I G N A L    P R O C E S S I N G    S U M M A R Y    R E S U L T S   ***');
    fprintf(Fout, '\n\n***   I N P U T S   ***');
    fprintf(Fout, '\n  CPU Operating Speed:   %s', DIG_SIG_TYPE);
    fprintf(Fout, '\n  Unit System:           %s', UNITS);

    fprintf(Fout, '\n\n*************************************************************************\n');

    %  Digital Signal Processing Properties
    fprintf(Fout, '\n\n** Sensor Properties **');
    fprintf(Fout, '\n  Digital Signal Processing Unit Type:       %s', DSP_MATRIX2{1,1});
    fprintf(Fout, '\n  CPU Operating Speed:                       %s', DSP_MATRIX2{1,2});
    fprintf(Fout, '\n  CPU Clock Speed:                        %12.3f   %s', DSP_MATRIX2{1,3},DSPUnitLabelStr{3} );
    fprintf(Fout, '\n  Number of CPU Processing Cores:         %12d   %s', DSP_MATRIX2{1,4},DSPUnitLabelStr{4} );
    fprintf(Fout, '\n  Amount of On-Board Data Memory:         %12.1f   %s', DSP_MATRIX2{1,5},DSPUnitLabelStr{5} );
    fprintf(Fout, '\n  Number of Data Bins:                    %12d   %s', DSP_MATRIX2{1,6},DSPUnitLabelStr{6} );
    fprintf(Fout, '\n  Amount of On-Board Program Memory:      %12.1f   %s', DSP_MATRIX2{1,7},DSPUnitLabelStr{7} );
    fprintf(Fout, '\n  Number of Memory Bins:                  %12d   %s', DSP_MATRIX2{1,8},DSPUnitLabelStr{8} );
    fprintf(Fout, '\n  Number of Computational Units:          %12d   %s', DSP_MATRIX2{1,9},DSPUnitLabelStr{9} );
    fprintf(Fout, '\n  CPU Cycle Time:                         %12.3f   %s', DSP_MATRIX2{1,10},DSPUnitLabelStr{10} );
    fprintf(Fout, '\n  Number of External Interrupts:          %12d   %s', DSP_MATRIX2{1,11},DSPUnitLabelStr{11} );


    fprintf(Fout, '\n\n  Power Requirements');
    fprintf(Fout, '\n  CPU Output Power:                       %12.3f   %s', DSP_MATRIX2{1,16},DSPUnitLabelStr{16} );
    fprintf(Fout, '\n  Input Voltage - Min:                    %12.3f   %s', DSP_MATRIX2{1,18},DSPUnitLabelStr{18} );
```

```
fprintf(Fout, '\n                    Max:                    %12.3f   %s', DSP_MATRIX2{1,19},DSPUnitLabelStr{19} );
fprintf(Fout, '\n  Voltage Type:                                          %s   %s',  DSP_MATRIX2{1,20},DSPUnitLabelStr{20} );
fprintf(Fout, '\n  Input Current:                          %12.3f   %s', DSP_MATRIX2{1,21},DSPUnitLabelStr{21} );


fprintf(Fout, '\n\n  Physical Properties');
fprintf(Fout, '\n  Processor Core Mass:                    %12.2f   %s', DSP_MATRIX2{1,17},DSPUnitLabelStr{17} );
fprintf(Fout, '\n  Sensor Dimensions - Length:             %12.3f   %s', DSP_MATRIX2{1,22},DSPUnitLabelStr{22} );
fprintf(Fout, '\n                        Width:            %12.3f   %s', DSP_MATRIX2{1,23},DSPUnitLabelStr{23} );
fprintf(Fout, '\n                       Height:            %12.3f   %s', DSP_MATRIX2{1,24},DSPUnitLabelStr{24} );


%  Environmental Properties
fprintf(Fout, '\n\n   Environmental Properties');
fprintf(Fout, '\n  Operating Temperature Range - Low:       %12.2f   %s', DSP_MATRIX2{1,12},DSPUnitLabelStr{12} );
fprintf(Fout, '\n                                High:       %12.2f   %s', DSP_MATRIX2{1,13},DSPUnitLabelStr{13} );
fprintf(Fout, '\n  Storage Temperature Range - Low:         %12.2f   %s', DSP_MATRIX2{1,14},DSPUnitLabelStr{14} );
fprintf(Fout, '\n                               High:       %12.2f   %s', DSP_MATRIX2{1,15},DSPUnitLabelStr{15} );

fprintf(Fout, '\n\n  Sensor Comments:');
for r = 1:length(DSP_MATRIX2{1,25})
    fprintf(Fout, '\n    %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
%  occured 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 a
%  erroneous input to the program and must be remedied by the user.
%
%
function ERROR_PRG(ERR_Code)

%  DISPLAY ERROR CODE NUMBER
fprintf('\n\n   *** 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  ***');
        %  Return to Main Directory
        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  ***');
        %  Return to Main Directory
        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  ***');
        %  Return to Main Directory
        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('   Unknown Unit Type entered into Input File. ');
        disp('   Verify UNIT is ''SI'' or ''British'' in Input File.');
        disp('***  TERMINATING ISSPO PROGRAM  ***');
        %  Return to Main Directory
        cd ..\..\..;
    case 51     %   CASE 51 ERROR UNIDENTIFIED PRINTFLG IN INPUT FILE
        disp('ERROR:   UNKNOWN PRINT FLAG IN INPUT FILE!');
        disp('   Unknown PRINTFLG variable entered into Input File. ');
        disp('   Verify PRINTFLG is ''Y'' or ''N'' in Input File.');
        disp('***  TERMINATING ISSPO PROGRAM  ***');
        %  Return to Main Directory
        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('   Program erred off in DataTypeVerifier Program. ');
        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:   UNDEFINED VOLUME_LIMIT VARIABLE IN INPUT FILE!');
        disp('   Program erred off in DataTypeVerifier Program. ');
        disp('   Verify VOLUME_LIMIT Variable is defined in Input File.');
        disp('***  TERMINATING ISSPO PROGRAM  ***');
        %  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 Requires Additional Design Information.');
        disp('   Verify Proper Data Input Format for each Sensor Type.');
        disp('***  TERMINATING ISSPO PROGRAM  ***');
        %  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 ..\..\..;
    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 ..\..\..;
    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 ..\..\..;
    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.');
```

```
        disp('   Specify Resolution as:  ''LOW''   ''MEDIUM''   ''HIGH''  ');
        disp('***  TERMINATING ISSPO PROGRAM  ***');
        %  Return to Main Directory
        cd ..\..\..;
    case 61    %   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('   Program erred off in DataTypeVerifier Program. ');
        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    %   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('   Program erred off in DataTypeVerifier Program. ');
        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    %   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('   Program erred off in DataTypeVerifier Program. ');
        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    %   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('   Program erred off in DataTypeVerifier Program. ');
        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    %   CASE 65 ERROR SENSOR OPTION INCLINATION REQUIRES RESOLUTION DATA IN
SENSOR_DATA
        disp('ERROR:   SENSOR OPTION INCLINATION REQUIRES ANGLE DATA IN SENSOR_DATA
ARRAY!');
        disp('   Program erred off in DataTypeVerifier Program. ');
        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    %   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('   Program erred off in DataTypeVerifier Program. ');
        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    %   CASE 67 ERROR SENSOR OPTION ACCELERATION REQUIRES RANGE 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 Range is defined in 2nd Column position.');
        disp('   Specify Range as:  ''SOFT''   ''MEDIUM''   ''HIGH''   ''IMPACT''
''BALLISTIC''  ');
        disp('***  TERMINATING ISSPO PROGRAM  ***');
        %  Return to Main Directory
        cd ..\..\..;
    case 68    %   CASE 68 ERROR SENSOR OPTION WIND VELOCITY 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 ''WIND VELOCITY'' Requires Additional Design
Information.');
        disp('   Verify Proper Data Input Format for each Sensor Type.');
        disp('***  TERMINATING ISSPO PROGRAM  ***');
        %  Return to Main Directory
        cd ..\..\..;
    case 69    %   CASE 69 ERROR SENSOR OPTION WIND VELOCITY 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 ''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
        cd ..\..\..;
    case 70    %   CASE 70 ERROR SENSOR OPTION GAS ANALYSIS REQUIRES ADDITIONAL DATA IN
SENSOR_DATA
        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
        cd ..\..\..;
    case 71    %   CASE 71 ERROR SENSOR OPTION GAS ANALYSIS REQUIRES ADDITIONAL DATA IN
SENSOR_DATA
        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
        cd ..\..\..;
    case 72    %   CASE 72 ERROR SENSOR OPTION GAS ANALYSIS REQUIRES ADDITIONAL DATA IN
SENSOR_DATA
        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
        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('   Program erred off in DataTypeVerifier Program. ');
        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''   ''Const Current''   ''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 Proper Data Input Format for each Sensor Type.');
        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 TEMPERATURE 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''  ');
        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 Proper Data Input Format for each Sensor Type.');
        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''  ');
        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
        disp('ERROR:   SENSOR OPTION DENSITY 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 Density Sensor type is defined in the next Column position.');
        disp('   Specify Type as:  ''GAS'' or ''LIQUID''  ');
        disp('***  TERMINATING ISSPO PROGRAM  ***');
        %  Return to Main Directory
        cd ..\..\..;
    case 81    %   CASE 81 ERROR OPTION DIGITAL SIGNAL PROCESSING REQUIRES ADDITIONAL
DATA IN SENSOR_DATA
        disp('ERROR:   OPTION DATA PROCESSING REQUIRES ADDITIONAL DATA IN SENSOR_DATA
ARRAY!');
        disp('   Program erred off in DataTypeVerifier Program. ');
        disp('   Selected Sensor Option ''DATA PROCESSING'' Requires Additional Design
Information.');
        disp('   Verify Proper Data Input Format for each Sensor Type.');
        disp('***  TERMINATING ISSPO PROGRAM  ***');
        %  Return to Main Directory
        cd ..\..\..;
    case 82    %   CASE 82 ERROR OPTION DIGITAL SIGNAL PROCESSING REQUIRES ADDITIONAL
DATA IN SENSOR_DATA
        disp('ERROR:   OPTION DATA PROCESSING REQUIRES ADDITIONAL DATA IN SENSOR_DATA
ARRAY!');
        disp('   Program erred off in DataTypeVerifier Program. ');
        disp('   Selected Sensor Option ''DATA PROCESSING'' Requires Additional Design
Information.');
        disp('   Verify CPU SPEED in 2nd Column Position, NUMBER OF DATA INPUTS in 3rd
Column.');
        disp('   Verify Proper Data Input Format for each Sensor Type.');
        disp('***  TERMINATING ISSPO PROGRAM  ***');
        %  Return to Main Directory
        cd ..\..\..;
    case 83    %   CASE 83 ERROR OPTION DIGITAL SIGNAL PROCESSING REQUIRES ADDITIONAL
DATA IN SENSOR_DATA
        disp('ERROR:   OPTION DATA PROCESSING REQUIRES ADDITIONAL DATA IN SENSOR_DATA
ARRAY!');
        disp('   Program erred off in DataTypeVerifier Program. ');
        disp('   Selected Sensor Option ''DATA PROCESSING'' Requires Additional Design
Information.');
        disp('   Verify CPU SPEED is defined in 2nd Column position.');
        disp('   Specify Range as:  ''LOW''   ''MEDIUM''   ''HIGH''  ');
        disp('***  TERMINATING ISSPO PROGRAM  ***');
        %  Return to Main Directory
        cd ..\..\..;
    case 84    %   CASE 84 ERROR OPTION DIGITAL SIGNAL PROCESSING REQUIRES ADDITIONAL
DATA IN SENSOR_DATA
        disp('ERROR:   OPTION DATA PROCESSING REQUIRES ADDITIONAL DATA IN SENSOR_DATA
ARRAY!');
        disp('   Program erred off in DataTypeVerifier Program. ');
        disp('   Selected Sensor Option ''DATA PROCESSING'' Requires Additional Design
Information.');
        disp('   Non-Numeric Number detected in SENSOR DATA Array 3rd Column.');
        disp('   Verify NUMBER OF DATA INPUTS expected is defined in 3rd Column
position.');
        disp('***  TERMINATING ISSPO PROGRAM  ***');
        %  Return to Main Directory
        cd ..\..\..;

    %   CASE 1XX SENSOR RESULT DATA FILES NOT CREATED PROPERLY
    case 100    %  ERROR 100 - TEMPERATURE SENSOR SUMMARY DATA FILE FAILED TO BE CREATED
        disp('ERROR:   TEMP SENSOR Database File Failed to be Created!');
        disp('   Program erred off in TEMP_SENSORS subroutine. ');
        disp('   Verify TEMP_SENSORS program is correctly creating the database file.');
        disp('***  TERMINATING ISSPO PROGRAM  ***');
        %  Return to Main Directory
```

```
        cd ..\..\..;
    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 ACCCELEROMETER_SENSORS subroutine. ');
        disp('   Verify ACCCELEROMETER_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 111    %  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 2XX SENSOR RESULT SUMMARY FILES NOT CREATED
    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 ..\..\..;
    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('***  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('***  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('***  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('***  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('***  TERMINATING ISSPO PROGRAM  ***');
        %  Return to Main Directory
        cd ..\..\..;

    otherwise
        disp('***   UNKNOWN ERROR DETECTED   ***');
        disp('Report ERROR and Case Configuration to Developer for Resolution!');
        fprintf('  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 ..\..\..;
```

D21. Mass Properties Program

```
%   function MASS_PROPERTIES(DataArray,UNITS)
%   Wind Velocity Sensor Database
%
%  Developed by:  Keith Schreck
%                 Mechanical and Aerospace Engineering
%                 San Jose State University
%  Date:          Fall 2007
%
%   This file contains the MASS PROPERTY calculation for the ISSPO program.
%  Data here is based on the sensors selected in the ISSPO Case Study and loads
%  the mass property values for weights and dimensions into a Mass Property
%  Array and calculates total values for all the components in the program.
%  Values are assigned to a variable name and saved to a MatLAB '.mat' file
%  and loaded when calling the ISSPO program.

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

%  Load Final sensor Matrix values into Main Program
k = 1;
for i = 1:size(DataArray,1)
    switch DataArray{i,1}
        case 'ATMOSPHERE'
            %   Loads and Converts Mass and Dimensional Properties to SI Standard Units For Comparison to Case Inputs
            if strcmpi(UNITS,'SI') == 1
                MASS_PROPS(k  ,1:4) = [ MATRIX2{1,5}/gr2kg          MATRIX2{1,6}/mm2m       MATRIX2{1,7}/mm2m
MATRIX2{1,8}/mm2m      ];
                MASS_PROPS(k+1,1:4) = [ P_MATRIX6{1,30}/gr2kg   P_MATRIX6{1,27}/mm2m   P_MATRIX6{1,28}/mm2m
P_MATRIX6{1,29}/mm2m  ];   %   Pressure Sensor - converts gr to kg and mm to meters
                if strcmpi(WV_TYPE,'ANEMOMETER') == 1
                    MASS_PROPS(k+2,1:4) = [ W_MATRIX3{1,12}         W_MATRIX3{1,16}/mm2m   W_MATRIX3{1,17}/mm2m
W_MATRIX3{1,18}/mm2m  ];   %  Probe Dimensions
                    MASS_PROPS2(1 ,1:4) = [       0                 W_MATRIX3{1,13}/mm2m   W_MATRIX3{1,14}/mm2m
W_MATRIX3{1,15}/mm2m  ];   %  Handheld Dimensions
                end
                if strcmpi(WV_TYPE,'DOPPLER') == 1
                    MASS_PROPS(k+2,1:4) = [ W_MATRIX3{1,24}         W_MATRIX3{1,25}/cm2m   W_MATRIX3{1,26}/cm2m
W_MATRIX3{1,27}/cm2m  ];   %  Probe Dimensions
                end
                MASS_PROPS(k+3,1:4) = [ H_MATRIX{1,18}/gr2kg    H_MATRIX{1,15}/mm2m    H_MATRIX{1,16}/mm2m
H_MATRIX{1,17}/mm2m   ];   %  Humidity Sensor
                MASS_PROPS(k+4,1:4) = [ D_MATRIX6{1,10}         D_MATRIX6{1,11}/mm2m   D_MATRIX6{1,12}/mm2m
D_MATRIX6{1,13}/mm2m  ];
                k = k + 5;
            end
            if strcmpi(UNITS,'British') == 1
                MASS_PROPS(k  ,1:4) = [ MATRIX2{1,5}/oz2lbm        MATRIX2{1,6}/in2ft      MATRIX2{1,7}/in2ft
MATRIX2{1,8}/in2ft     ];
                MASS_PROPS(k+1,1:4) = [ P_MATRIX6{1,30}/oz2lbm    P_MATRIX6{1,27}/in2ft  P_MATRIX6{1,28}/in2ft
P_MATRIX6{1,29}/in2ft  ];   %   Pressure Sensor
                if strcmpi(WV_TYPE,'ANEMOMETER') == 1
                    MASS_PROPS(k+2,1:4) = [ W_MATRIX3{1,12}           W_MATRIX3{1,16}/in2ft  W_MATRIX3{1,17}/in2ft
W_MATRIX3{1,18}/in2ft  ];   %  Probe Dimensions
                    MASS_PROPS2(1 ,1:4) = [       0                   W_MATRIX3{1,13}/in2ft  W_MATRIX3{1,14}/in2ft
W_MATRIX3{1,15}/in2ft  ];   %  Handheld Dimensions
                end
                if strcmpi(WV_TYPE,'DOPPLER') == 1
                    MASS_PROPS(k+2,1:4) = [ W_MATRIX3{1,24}         W_MATRIX3{1,25}/in2ft   W_MATRIX3{1,26}/in2ft
W_MATRIX3{1,27}/in2ft  ];   %  Probe Dimensions
                end
                MASS_PROPS(k+3,1:4) = [ H_MATRIX{1,18}/oz2lbm     H_MATRIX{1,15}/in2ft   H_MATRIX{1,16}/in2ft
H_MATRIX{1,17}/in2ft   ];   %  Humidity Sensor
                MASS_PROPS(k+4,1:4) = [ D_MATRIX6{1,10}           D_MATRIX6{1,11}/in2ft  D_MATRIX6{1,12}/in2ft
D_MATRIX6{1,13}/in2ft  ];
                k = k + 5;
            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
ACC_MATRIX6{1,23}/cm2m   ];
```

```
            k = k + 1;
        end
        if strcmpi(UNITS,'British') == 1
            MASS_PROPS(k  ,1:4) = [ ACC_MATRIX6{1,20}/oz2lbm  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   ];
            k = k + 1;
        end
        if strcmpi(UNITS,'British') == 1
            MASS_PROPS(k  ,1:4) = [ ACS_MATRIX4{1,28}/oz2lbm   ACS_MATRIX4{1,25}/in2ft   ACS_MATRIX4{1,26}/in2ft
ACS_MATRIX4{1,27}/in2ft  ];
            k = k + 1;
        end
    case 'EM FIELD'                                                                                    %  EM
FIELD Sensor
        if strcmpi(UNITS,'SI') == 1
            MASS_PROPS(k  ,1:4) = [ AC_MATRIX5{1,23}/gr2kg     AC_MATRIX5{1,17}/cm2m     AC_MATRIX5{1,18}/cm2m
AC_MATRIX5{1,19}/cm2m  ];   %  Probe Dimensions
            MASS_PROPS3(1 ,1:4) = [ AC_MATRIX5{1,24}/gr2kg     AC_MATRIX5{1,20}/cm2m     AC_MATRIX5{1,21}/cm2m
AC_MATRIX5{1,22}/cm2m  ];   %  Handheld Dimensions
            k = k + 1;
        end
        if strcmpi(UNITS,'British') == 1
            MASS_PROPS(k  ,1:4) = [ AC_MATRIX5{1,23}/oz2lbm    AC_MATRIX5{1,17}/in2ft    AC_MATRIX5{1,18}/in2ft
AC_MATRIX5{1,19}/in2ft  ];   %  Probe Dimensions
            MASS_PROPS3(1 ,1:4) = [ AC_MATRIX5{1,24}/oz2lbm    AC_MATRIX5{1,20}/in2ft    AC_MATRIX5{1,21}/in2ft
AC_MATRIX5{1,22}/in2ft  ];   %  Handheld Dimensions
            k = k + 1;
        end
    case 'GAS ANALYSIS'
        if strcmpi(UNITS,'SI') == 1
            if strcmpi(GCMS_TYPE,'WAVELENGTH') == 1
                MASS_PROPS(k  ,1:4) = [ GCMS_MATRIX3{1,8}    GCMS_MATRIX3{1,9}/mm2m    GCMS_MATRIX3{1,10}/mm2m
GCMS_MATRIX3{1,11}/mm2m  ];
                k = k + 1;
            end
            if strcmpi(GCMS_TYPE,'MASS-CHARGE') == 1
                MASS_PROPS(k  ,1:4) = [ GCMS_MATRIX3{1,33}    GCMS_MATRIX3{1,25}/cm2m    GCMS_MATRIX3{1,26}/cm2m
GCMS_MATRIX3{1,27}/cm2m  ];
                k = k + 1;
            end
        end
        if strcmpi(UNITS,'British') == 1
            if strcmpi(GCMS_TYPE,'WAVELENGTH') == 1
                MASS_PROPS(k  ,1:4) = [ GCMS_MATRIX3{1,8}   GCMS_MATRIX3{1,9}/in2ft   GCMS_MATRIX3{1,10}/in2ft
GCMS_MATRIX3{1,11}/in2ft  ];
                k = k + 1;
            end
            if strcmpi(GCMS_TYPE,'MASS-CHARGE') == 1
```

```
                    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
            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}/oz2lbm  INCL_MATRIX5{1,16}/in2ft  INCL_MATRIX5{1,17}/in2ft
INCL_MATRIX5{1,18}/in2ft  ];
                k = k + 1;
            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
];
                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
];
                k = k + 1;
            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
        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}/oz2lbm   RAD_MATRIX3{1,27}/in2ft   RAD_MATRIX3{1,28}/in2ft
RAD_MATRIX3{1,29}/in2ft ];
                k = k + 1;
            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}/oz2lbm   REF_MATRIX3{1,12}/in2ft   REF_MATRIX3{1,13}/in2ft
REF_MATRIX3{1,14}/in2ft ];
                k = k + 1;
            end
        case 'DATA PROCESSING'
            if strcmpi(UNITS,'SI') == 1
                MASS_PROPS(k  ,1:4) = [ DSP_MATRIX2{1,17}   DSP_MATRIX2{1,22}/mm2m   DSP_MATRIX2{1,23}/mm2m
DSP_MATRIX2{1,24}/mm2m ];
                k = k + 1;
            end
            if strcmpi(UNITS,'British') == 1
                MASS_PROPS(k  ,1:4) = [ DSP_MATRIX2{1,17}   DSP_MATRIX2{1,22}/in2ft   DSP_MATRIX2{1,23}/in2ft
DSP_MATRIX2{1,24}/in2ft ];
                k = k + 1;
            end
        otherwise
    end
end

save('MASS_PROPERTIES_Data.mat');
```


D22. Power Properties Program

```
%   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 * I in Watts
%

function POWER_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


%  Load Final sensor Matrix values into Main Program
k = 1;
for i = 1:size(DataArray,1)
    switch DataArray{i,1}
        case 'ATMOSPHERE'
            %   Loads and Converts POWER Properties to SI Standard Units For Comparison to Case Inputs
            if strcmpi(UNITS,'SI') == 1
                POWER_PROPS(k  ,1) = [ MATRIX2{1,4}     ];
                POWER_PROPS(k+1,1) = [ P_MATRIX6{1,22} * P_MATRIX6{1,23}/mA2A  ];   %   Pressure Sensor - converts  V & mA to W
                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  ]; % - 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{1,22} * P_MATRIX6{1,23}/mA2A * BTU_hr2W  ];   %   Pressure Sensor- converts  V
& mA to W to BTU/hr
                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;
            end
        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{1,15} * ACC_MATRIX6{1,14}/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{1,15} * ACC_MATRIX6{1,14}/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_PROPS3(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(GCMS_TYPE,'WAVELENGTH') == 1
            POWER_PROPS(k  ,1) = [         0        ];  %  Voltage Only no Current Data
            k = k + 1;
        end
        if strcmpi(GCMS_TYPE,'MASS-CHARGE') == 1
            POWER_PROPS(k  ,1) = [  GCMS_MATRIX3{1,37}  ];
            k = k + 1;
        end
    end
    if strcmpi(UNITS,'British') == 1
        if strcmpi(GCMS_TYPE,'WAVELENGTH') == 1
            POWER_PROPS(k  ,1) = [         0        ];  %  Voltage Only no Current Data
            k = k + 1;
        end
        if strcmpi(GCMS_TYPE,'MASS-CHARGE') == 1
            POWER_PROPS(k  ,1) = [  GCMS_MATRIX3{1,37}  ];
            k = k + 1;
        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
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
case 'OPTICS'
    if strcmpi(UNITS,'SI') == 1
        POWER_PROPS(k  ,1) = [   IMG_MATRIX5{1,30}/mW2W   ];   %  mW to W
        k = k + 1;
    end
    if strcmpi(UNITS,'British') == 1
        POWER_PROPS(k  ,1) = [   IMG_MATRIX5{1,30}   ];   %
        k = k + 1;
    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
        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
        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
        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.

ISSPO Program ERROR Codes

PROGRAM OPERATION ERRORS		
ERROR CODE	**SOURCE MODULE**	**ACTION**
1	ISSPO	INPUT_FILE does not exist. Verify correct input file name.
2	ISSPO	InputDeck file does not exist. InputDeck file not successfully created.
3	CONSTANTS	CONSTANTS database file not created. Verify Constants database is being created.
4	PLANETARY_ DATABASE	Planetary database file does not exist. Verify the Planetary Database program is correctly making the database file.
UNKNOWN PROGRAM VARIABLES		
ERROR CODE	**SOURCE MODULE**	**ACTION**
50	ISSPO	Invalid UNITS variable entered into input file. Verify UNITS is 'SI' or 'British'.
51	ISSPO	Invalid Print Option variable entered into input file. Verify PRINTFLG is 'Y' or 'N'. Prints out individual program results to main program window.
52	DATA_TYPE_ VERIFIER	Invalid type of science data entered into SENSOR_DATA Array. Verify sensor data types entered into array are valid science data options.
53	ISSPO	Undefined Mass Limit variable entered into input file. Verify MASS_LIMIT variable is set and defined in user input file.
54	ISSPO	Undefined Power Limit variable entered into input file. Verify POWER_LIMIT variable is set and defined in user input file.
55	ISSPO	Undefined Volume Limit variable entered into input file. Verify VOLUME_LIMIT variable is set and defined in user input file.
56	DATA_TYPE_ VERIFIER	Sensor Option Requires additional Input parameters in SENSOR_DATA Variable in Input file. Verify proper data format for selected sensor.

57	DATA_TYPE_ VERIFIER	Sensor Option 'OPTICS' Requires additional Input parameters in SENSOR_DATA Variable in Input file. Verify proper data format for selected sensor. Size of SENSOR_DATA = 2
58	DATA_TYPE_ VERIFIER	Sensor Option 'OPTICS' Requires additional Input parameters in SENSOR_DATA Variable in Input file. Verify proper data format for selected sensor. Size of SENSOR_DATA = 3
59	DATA_TYPE_ VERIFIER	Invalid value in sensor option 'OPTICS'. Requires Optical Sensor System Type in 2nd column position in SENSOR_DATA variable in Input file. Valid Options are 'CAMERA, 'ARRAY', 'LINEAR'. Verify proper system format for selected sensor.
60	DATA_TYPE_ VERIFIER	Invalid value in sensor option 'OPTICS'. Requires Optical Sensor Resolution in 3rd column position in SENSOR_DATA variable in Input file. Valid Resolutions are 'LOW', 'MEDIUM', 'HIGH'. Verify proper resolution format for selected sensor.
61	DATA_TYPE_ VERIFIER	Invalid value in sensor option 'OPTICS'. Requires Optical Sensor Type in 4th column position in SENSOR_DATA variable in Input file. Valid image types are 'X-RAY', 'VISUAL', 'UV', 'NIR', and 'MICRO'. Verify proper image format for selected sensor. Multiple Image types are allowed. Enter additional image types in subsequent columns.
62	DATA_TYPE_ VERIFIER	Sensor Option 'RADIATION' Requires additional Input parameters in SENSOR_DATA Variable in Input file. Verify proper data format for selected sensor.
63	DATA_TYPE_ VERIFIER	Invalid value in sensor option 'RADIATION'. Requires Radiation Sensor Type in 3rd column position in SENSOR_DATA variable in Input file. Valid radiation types are 'Charged Particle', 'Alpha', 'Beta', 'Gamma', and 'X-Ray'. Verify proper radiation type for selected sensor. Multiple Radiation types are allowed. Enter additional radiation types in subsequent columns.
64	DATA_TYPE_ VERIFIER	Sensor Option 'INCLINATION' Requires additional Input parameters in SENSOR_DATA Variable in Input file. Verify proper data format for selected sensor.
65	DATA_TYPE_	Invalid value in sensor option

	VERIFIER	'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.
66	DATA_TYPE_ VERIFIER	Sensor Option 'ACCELERATION' Requires additional Input parameters in SENSOR_DATA Variable in Input file. Verify proper data format for selected sensor.
67	DATA_TYPE_ VERIFIER	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.
68	DATA_TYPE_ VERIFIER	Sensor Option 'ATMOSPHERE' – 'WIND VELOCITY' Requires additional Input parameters in SENSOR_DATA Variable in Input file. Verify proper data format for selected sensor.
69	DATA_TYPE_ VERIFIER	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.
70	DATA_TYPE_ VERIFIER	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.
71	DATA_TYPE_ VERIFIER	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. Verify proper system format for selected sensor.
72	DATA_TYPE_ VERIFIER	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.
73	DATA_TYPE_ VERIFIER	Sensor Option 'ACCELERATION' Requires additional Input parameters in

		SENSOR_DATA Variable in Input file. Sensor Power Type value required in 3rd Column. Verify proper data format for selected sensor.
74	DATA_TYPE_ VERIFIER	Invalid value in sensor option 'ACCELERATION'. Requires Acceleration Sensor Power Type in 3rd column position in SENSOR_DATA variable in Input file. Valid types are 'Voltage', 'Const Currant', 'Self Generating'. Verify proper range format for selected sensor.
75	DATA_TYPE_ VERIFIER	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.
76	DATA_TYPE_ VERIFIER	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.
77	DATA_TYPE_ VERIFIER	Sensor Option 'ACOUSTICS' Requires additional Input parameters in SENSOR_DATA Variable in Input file. Verify proper data format for selected sensor.
78	DATA_TYPE_ VERIFIER	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.
79	DATA_TYPE_ VERIFIER	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.
80	DATA_TYPE_ VERIFIER	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.
81	DATA_TYPE_ VERIFIER	Sensor Option 'DATA PROCESSING' Requires additional Input parameters in SENSOR_DATA Variable in Input file. Verify proper data format for selected

		sensor.
82	DATA_TYPE_ VERIFIER	Sensor Option 'DATA PROCESSING' Requires additional Input parameters in SENSOR_DATA Variable in Input file. Requires Microprocessor Cycle speed in 2nd column position and number of connected devices in 3rd column position. Verify proper data format for selected sensor.
83	DATA_TYPE_ VERIFIER	Invalid value in sensor option 'DATA PROCESSING'. Requires Microprocessor Cycle speed in 2nd column position in SENSOR_DATA variable in Input file. Valid types are 'LOW', 'MEDIUM', or 'HIGH'. Verify proper operating medium for selected sensor.
84	DATA_TYPE_ VERIFIER	Invalid value in sensor option 'DATA PROCESSING'. Required number of connected devices in SENSOR_DATA variable in Input file. Non-numeric value detected in 3rd column position. Verify proper operating medium for selected sensor.
SENSOR DATA FILES		
ERROR CODE	**SOURCE MODULE**	**ACTION**
100	TEMP_SENSORS	TEMP_SENSORS Database file was not created successfully. Program erred off somewhere in module.
101	PRESSURE_ SENSORS	PRESSURE_SENSORS Database file was not created successfully. Program erred off somewhere in module.
102	NEPHELOMETER_ SENSOR	NEPHELOMETER_SENSORS Database file was not created successfully. Program erred off somewhere in module.
103	HUMIDITY_ SENSORS	HUMIDITY_SENSORS Database file was not created successfully. Program erred off somewhere in module.
104	DENSITY_ SENSORS	DENSITY_SENSORS Database file was not created successfully. Program erred off somewhere in module.
105	ACOUSTIC_ SENSORS	ACOUSTIC_SENSORS Database file was not created successfully. Program erred off somewhere in module.
106	AC_FIELD_ SENSORS	AC_FIELD_SENSORS Database file was not created successfully. Program erred off somewhere in module.
107	WIND_VELOCITY_ SENSORS	WIND_VELOCITY_SENSORS Database file was not created successfully. Program erred off somewhere in module.
108	ACCELEROMETER_	ACCELEROMETER_SENSORS Database

	SENSORS	file was not created successfully. Program erred off somewhere in module.
109	GCMS_ SENSORS	GCMS_SENSORS Database file was not created successfully. Program erred off somewhere in module.
110	RADITATION_ SENSORS	RADIATION_SENSORS Database file was not created successfully. Program erred off somewhere in module.
111	INCLINOMETER_ SENSORS	INCLINOMETER_SENSORS Database file was not created successfully. Program erred off somewhere in module.
112	IMAGING_ SENSORS	IMAGING_SENSORS Database file was not created successfully. Program erred off somewhere in module.
113	REFRACTION_SENSORS	REFRACTION_SENSORS Database file was not created successfully. Program erred off somewhere in module.
114	DIGITAL_SIGNAL_ PROCESSING	DIGITAL_SIGNAL_PROCESSING Database file was not created successfully. Program erred off somewhere in module.
SENSOR SUMMARY FILES		
ERROR CODE	**SOURCE MODULE**	**ACTION**
200	TEMP_SENSORS	TEMP_SENSORS Summary file was not created successfully. Program erred off somewhere in module.
201	PRESSURE_ SENSORS	PRESSURE_SENSORS Summary file was not created successfully. Program erred off somewhere in module.
202	NEPHELOMETER_ SENSOR	NEPHELOMETER_SENSORS Summary file was not created successfully. Program erred off somewhere in module.
203	HUMIDITY_ SENSORS	HUMIDITY_SENSORS Summary file was not created successfully. Program erred off somewhere in module.
204	DENSITY_ SENSORS	DENSITY_SENSORS Summary file was not created successfully. Program erred off somewhere in module.
205	ACOUSTIC_ SENSORS	ACOUSTIC_SENSORS Summary file was not created successfully. Program erred off somewhere in module.
206	AC_FIELD_ SENSORS	AC_FIELD_SENSORS Summary file was not created successfully. Program erred off somewhere in module.
207	WIND_VELOCITY_ SENSORS	WIND_VELOCITY_SENSORS Summary file was not created successfully. Program erred off somewhere in module.
208	ACCELEROMETER_ SENSORS	ACCELEROMETER_SENSORS Summary file was not created successfully. Program erred off somewhere in module.

209	GCMS_ SENSORS	GCMS_SENSORS Summary file was not created successfully. Program erred off somewhere in module.
210	RADITATION_ SENSORS	RADIATION_SENSORS Summary file was not created successfully. Program erred off somewhere in module.
211	INCLINOMETER_ SENSORS	INCLINOMETER_SENSORS Summary file was not created successfully. Program erred off somewhere in module.
212	IMAGING_ SENSORS	IMAGING_SENSORS Summary file was not created successfully. Program erred off somewhere in module.
213	REFRACTION_ SENSORS	REFRACTION_SENSORS Summary file was not created successfully. Program erred off somewhere in module.
214	DIGITAL_SIGNAL_ PROCESSING	DIGITAL_SIGNAL_PROCESSING Summary file was not created successfully. Program erred off somewhere in module.
215	MASS_ PROPERTIES	MASS_PROPERTIES SUMMARY file was not created successfully. Program erred off somewhere in module.
216	POWER_ PROPERTIES	POWER_PROPERTIES SUMMARY file was not created successfully. Program erred off somewhere in module.