NASA/TM-1998-208466

Designing for Change: Minimizing the Impact of Changing Requirements in the Later Stages of a Spaceflight Software Project

B. Danette Allen
Langley Research Center, Hampton, Virginia

October 1998

The NASA STI Program Office ... in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) Program Office plays a key part in helping NASA maintain this important role.

The NASA STI Program Office is operated by Langley Research Center, the lead center for NASA's scientific and technical information. The NASA STI Program Office provides access to the NASA STI Database, the largest collection of aeronautical and space science STI in the world. The Program Office is also NASA's institutional mechanism for disseminating the results of its research and development activities. These results are published by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counter-part of peer reviewed formal professional papers, but having less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services that help round out the STI Program Office's diverse offerings include creating custom thesauri, building customized databases, organizing and publishing research results ... even providing videos.

For more information about the NASA STI Program Office, see the following:

- Access the NASA STI Program Home Page at ***http://www.sti.nasa.gov***

- E-mail your question via the Internet to help@sti.nasa.gov

- Fax your question to the NASA Access Help Desk at (301) 621-0134

- Phone the NASA Access Help Desk at (301) 621-0390

- Write to:
  NASA Access Help Desk
  NASA Center for AeroSpace Information
  7121 Standard Drive
  Hanover, MD 21076-1320

NASA/TM-1998-208466

Designing for Change: Minimizing the Impact of Changing Requirements in the Later Stages of a Spaceflight Software Project

B. Danette Allen
Langley Research Center, Hampton, Virginia

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

October 1998

Available from the following:

NASA Center for AeroSpace Information (CASI)
7121 Standard Drive
Hanover, MD 21076-1320
(301) 621-0390

National Technical Information Service (NTIS)
5285 Port Royal Road
Springfield, VA 22161-2171
(703) 487-4650

Designing for Change: Minimizing the Impact of Changing Requirements in the Later Stages of a Spaceflight Software Project

B. Danette Allen
NASA Langley Research Center
M/S 488
Hampton, VA 23681-0001
b.d.allen@larc.nasa.gov

Abstract

In the traditional "waterfall" model of the software project life cycle, the Requirements Phase ends and flows into the Design Phase, which ends and flows into the Development Phase. Unfortunately, the process rarely, if ever, works so smoothly in practice. Instead, software developers often receive new requirements, or modifications to the original requirements, well after the earlier project phases have been completed. In particular, projects with shorter than ideal schedules are highly susceptible to frequent requirements changes, as the software requirements analysis phase is often forced to begin before the overall system requirements and top-level design are complete. This results in later modifications to the software requirements, even though the software design and development phases may be complete. Requirements changes received in the later stages of a software project inevitably lead to modification of existing developed software. Presented here is a series of software design techniques that can greatly reduce the impact of last-minute requirements changes. These techniques were successfully used to add built-in flexibility to two complex software systems in which the requirements were expected to (and did) change frequently. These large, real-time systems were developed at NASA Langley Research Center (LaRC) to test and control the Lidar In-Space Technology Experiment (LITE) instrument which flew aboard the space shuttle Discovery as the primary payload on the STS-64 mission. Although developed in the Ada programming language, most of these methods are language-independent and can easily be modified to be used in any high-level language.

Acronym List

AT	IBM PC AT
CAP	Command Acceptance Pattern
CDR	Critical Design Review
CFG	Command File Generator
DOD	Department of Defense
GSE	Ground Support Equipment
IC	Instrument Controller
I&T	Integration and Test
JSC	Johnson Space Center
KSC	Kennedy Space Center
LaRC	Langley Research Center
Lidar	Light Detection and Ranging
LITE	Lidar In-space Technology Experiment
MSFC	Marshall Space Flight Center
NASCOM	NASA Communications
OOD	Object Oriented Design
PIC	Payload Interface Converter
PGSC	Payload General Support Computer
POCC	Payload Operations Control Computer
SFMDM	Smart Flexible Multiplexer DeMultiplexer
SRS	System Requirements Specification
STS	Space Transportation System

1. Introduction

The Lidar In-space Technology Experiment (LITE) is a three-wavelength backscatter lidar developed by NASA Langley Research Center (LaRC) that flew on the space shuttle Discovery in September 1994 as primary payload on the STS-64 mission. Lidar (Light Detection ad Ranging) is a remote sensing technique that can be used to study clouds ad aerosols in the atmosphere by sending out short pulses of laserlight and detecting the portion scattered back to the instrument. The goals of the LITE mission were to validate key lidar technologies for spaceborne applications, to explore the applications of space lidar, and to gain operational experience which will benefit the development of future systems on free-flying satellite platforms. The 40 Gbytes of science data collected provide the first highly detailed global view of the vertical structure of clouds and aerosols from the Earth's surface through the middle stratosphere. LITE also provided the first demonstration of space lidar for study of the Earth's atmosphere and resulted in the collection of 53 hours of data, equivalent to 35 orbits of the Earth. This is the first data set that provides the highly detailed vertical structure of clouds and aerosols on a global scale. The success of the mission demonstrates the maturity of lidar technology and the potential for long-term orbital lidars on free-flying platforms. The LITE mission is detailed further in [1].

Two sets of real-time Ground Support Equipment (GSE) software were required for the development, integration, test and flight of the LITE payload. Integration and test required a real-time platform that simulated absent shuttle interfaces, displayed engineering and science data and provided a means for commanding the instrument. Mission software was required for use in the Payload Operations Control Computer (POCC) facility for commanding and data display during the flight. The software team was assembled with 18 months to develop and deliver the test GSE and, following a successful test at Kennedy Space Center (KSC), 12 months to develop and deliver mission critical ground software to Johnson Space Center (JSC) for integration into the POCC. The test GSE was developed in parallel with the instrument and was required to simulate yet-to-be-defined interfaces. The mission software would follow-on from the test software but would require major changes to accommodate new interfaces onboard the shuttle and at JSC. Both systems were developed on an accelerated schedule by small teams of three to five people. We knew we had to design for change. In order to design for change, we needed to predict what could change and when. This required that we understand why software requirements changes occur. Five change agents were identified: immature system requirements, documentation, detail comprehension, shifting user requirements and external influences. As immature external requirements were defined and new ones added, our design needed to minimize impact on the existing system. The decision was made to counter risk by driving the uncertainty into the data, as opposed to the code, through the use of external files.

The following sections will describe techniques used on these systems to build in flexibility in order to minimize the impact of late requirements changes on the software systems. A history of the LITE payload is provided for context. The disconnect between academic life cycle models and their application in our working environment is presented

along with our chosen data-driven approach. Finally, the philosophies and techniques that were successfully used in designing for change are discussed in detail.

2. LITE History

During the 10-day mission, commanding of the LITE instrument was accomplished primarily from Mission Control at JSC in Houston. All commands are received by the on-board Instrument Controller (IC), which parses the commands and relays them to the Aft Optics or Boresight Assembly subsystems, if required. The IC executes instrument level commands. The LITE instrument command set is very versatile, including over 200 commands to control every facet of the instrument operation. The LITE Command File Generator (CFG) operated without error for the entire mission duration.

The CFG is a relatively large (over 30,000 Ada statements), PC-based real-time platform from which all commands were issued to the LITE instrument during its 10-day mission. It was developed specifically for LITE and allowed for commanding the IC directly or via the Payload General Support Computer (PGSC) in the shuttle flight deck. The CFG requirements [2] are summarized in Table 1.

	Requirement	**Center**	**Reference**
1	Display relevant Payload Data Interleaver (PDI) and LITE data	LaRC JSC	
2	Retrieve instrument data from Lantastics network	LaRC	
3	Conform to NASA Communication (NASCOM) uplink format	JSC	[3]
4	Conform to Smart Flexible Multiplexer Demultiplexer command format	MSFC	[5]
5	Conform to LITE IC command format	LaRC	[4]
6	Conform to PGSC command format	MSFC	[5]
7	Interface directly to Payload Interface Converter (PIC) for uplink commands	JSC	[3]
8	Receive Command Acceptance Pattern (CAP) from PIC	JSC	[3]

Table 1: CFG Requirements

As shown in Table 1, requirements from numerous NASA centers and, therefore, numerous documents, had to be understood and merged in order to generate the CFG requirements and platform. Figure 1 shows the CFG data path.

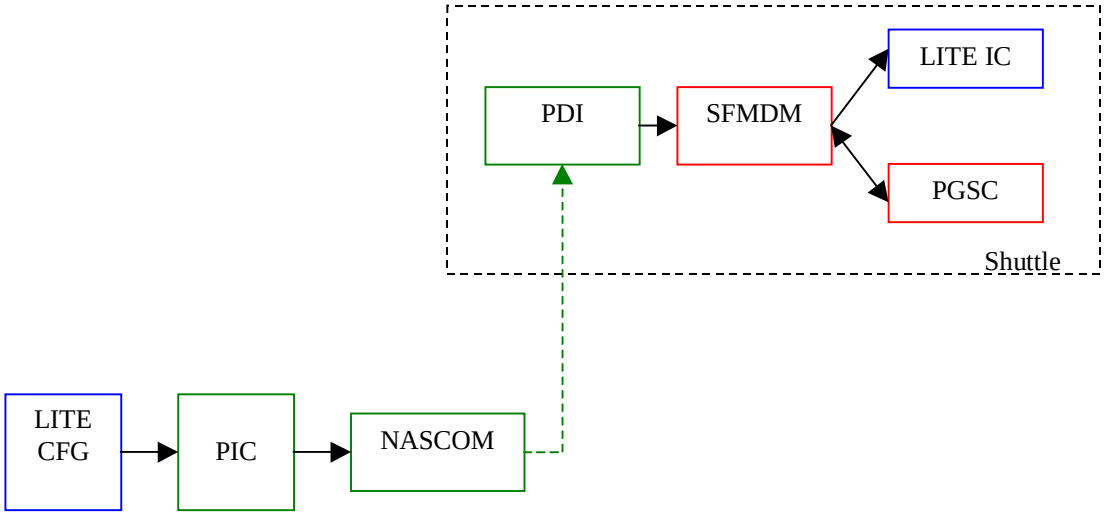

Figure 1: LITE CFG Data Path

Before any resources could be put towards the design, development and delivery of a mission commanding platform, test GSE was needed for integration and test of LITE at LaRC and KSC. Software was brought in late in the LITE project lifecycle. Any and all problems that had been hand-waved off with "we'll take care of that in software" had to be addressed. The IC was the heart of the LITE instrument and had no formal requirements. The LITE instrument needed to be fully integrated and tested in 18 months. Test GSE development would have to be in parallel with the design of the IC software and integration of the instrument.

The Pallet Simulator (Figure 2) is a large (over 60,000 Ada statements), real-time, distributed system consisting of three platforms: Master AT, Display AT, and Quicklook AT. The Master AT interfaced directly with the LITE instrument via in-house GSE. It simulated a shuttle interface, the Smart Flexible Multiplexer Demultiplexer (SFMDM) [5], through which all commands and engineering data were transferred. For test purposes, it also provided the instrument interface for science data, again though in-house GSE. Because this platform simulated a shuttle interface, the real-time requirements of the system were its first priority and could not be compromised. These requirements were successfully married with the storage of science and engineering data along with the transfer of data to the appropriate platform, either the Quicklook AT or the Display AT. These two platforms were receive-only except for communication health checks with the Master AT and, therefore, did not interface directly with the instrument. For this reason, discussion of the Pallet simulator will be limited to the Master AT software only.

The design and development of the Master AT (IBM PC AT) software coincided with the design, integration and test of the LITE instrument. Therefore, requirements changes were inevitable. The team was forced to devise an approach that would allow for built-in system flexibility. Because the instrument was still at the subsystem level, the IC was still

at the conceptual stage. Instrument commanding and data handling were undefined. We needed autonomy but did not have an operations scenario. Little thought had been given to exactly how these items would be accomplished. What would a LITE command look like? What was the downlink data format? We had the challenge of developing a test system for an instrument that was, from a software perspective, undefined. The flight software and GSE software would be defined, designed, developed and tested in parallel alongside the instrument itself.

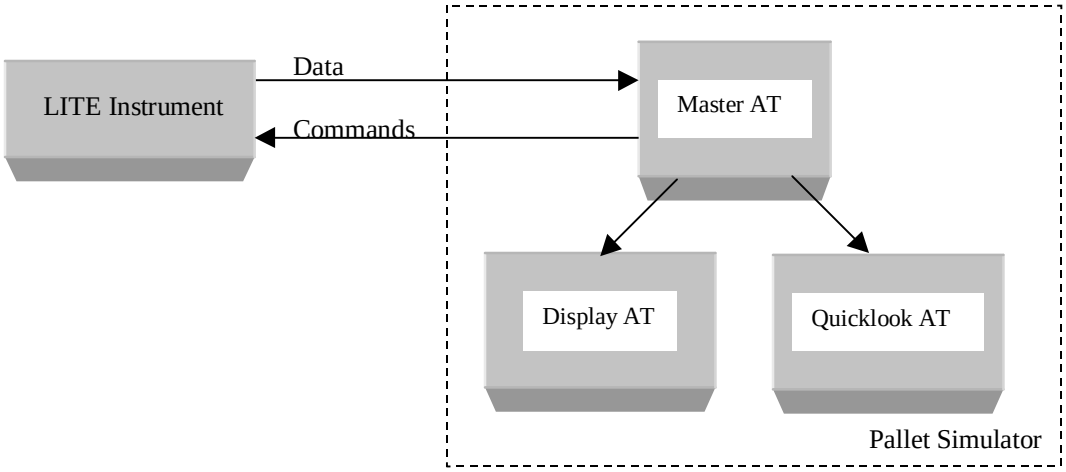

Figure 2: Master AT Architecture

The software team was faced with a challenge. The CFG would follow-on from the Master AT. The Master AT had to be delivered coincident with the semi-specified LITE instrument and the unspecified LITE IC. Figure 3 illustrates the major milestones in the LITE project timeline. Each box marks the beginning of the specified task but is not intended to communicate task duration. Of particular interest, the final System Requirement Specification (SRS) was not available until the Master AT software was delivered and the CFG software was in development.

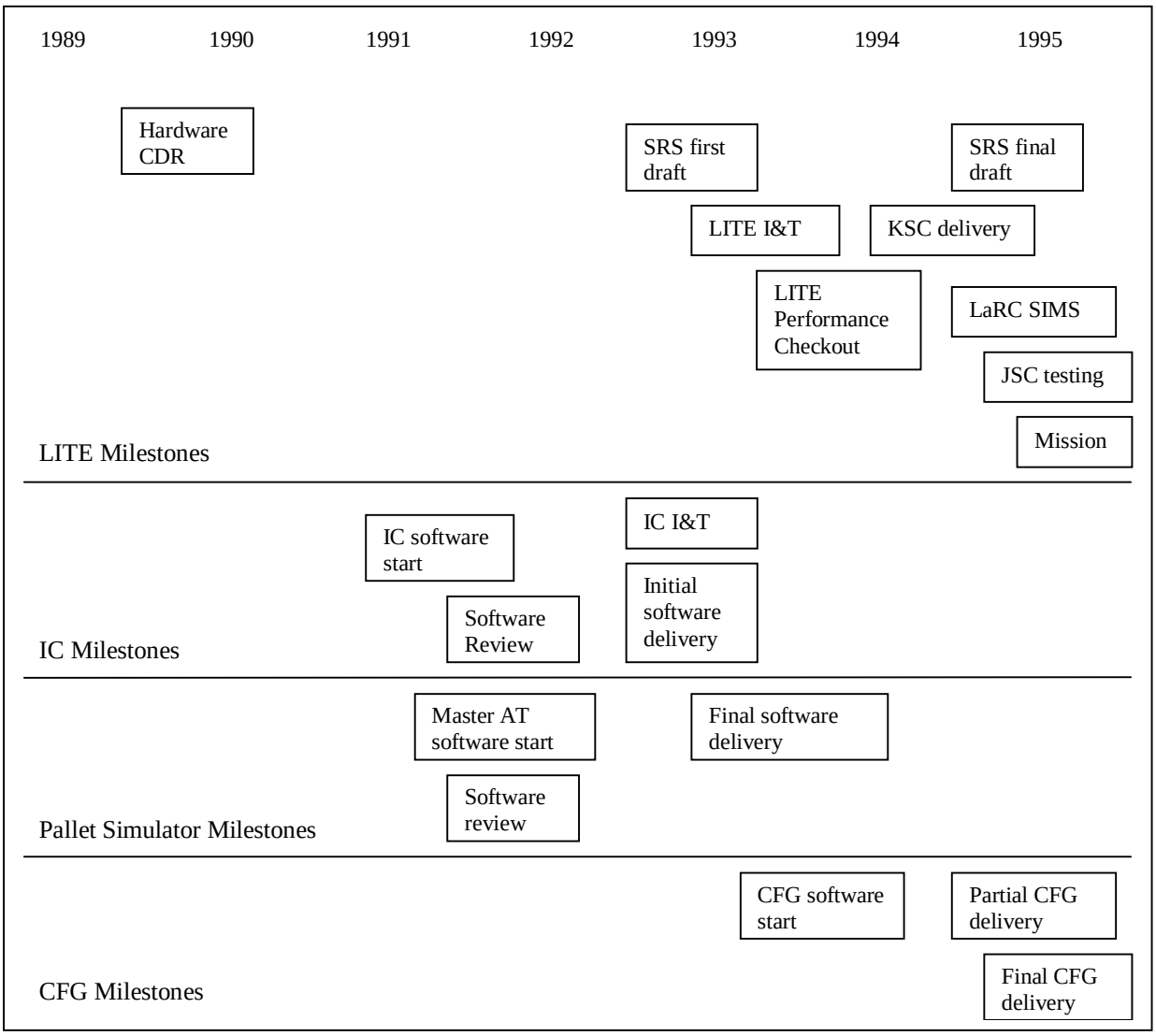

Figure 3: LITE Project Major Milestones

We needed to design and code to minimize system impacts when changes occurred. We also needed to design with reuse in mind knowing that the CFG would leverage off the Master AT development. The traditional waterfall model where all activity of a certain type occurs during a phase of the same name and in which phases do not overlap [6], was simply not an appropriate model. The spiral model requires several revolutions requiring time and experience, neither of which we had in abundance. We needed an approach suitable for the accelerated schedule under which we were to develop mission-critical GSE. Prototyping seemed the best choice but was not entirely appropriate given our short schedule and real time requirements. We combined the appropriate philosophies from several models in creating a data-driven approach that was workable within our short development schedule. We also needed to design for change. The area most susceptible to change was the data so we countered risk by using a data-driven design that generated code to accommodate changes in external text files. Presented here are the design and coding techniques used in the development of these complex platforms.

3. The Unstable Process of Software Design and Development

Change is a very natural and intrinsic aspect of the software development process. For example, change occurs [7]:

- during software requirements analysis phase, when the software requirements specification is agreed to by all parties and is baselined.
- during software development and test, when the software design, or the code, or the test plans are completed.
- during software maintenance, when requirements have changed or matured, and
- during software maintenance or later enhancement, when a problem is detected in the software requirements or system that forces modifications to the design or the code or the test plan.

Knowing that software will change is not enough. In order to design for change, one must understand why software will change.

3.1 Immature System Requirements

The trend to implement in software functions traditionally accomplished by hardware devices has led to increased size and complexity of software subsystems [18]. Historically, software has been a very small part of the total project cost and was managed as such. Older paradigms provide little or no guidance for managing software projects [8]. As a result, the need for software often goes unrecognized at the beginning stages of a project. Even once the need for software is recognized, the resources required are often underestimated, especially in the early requirements specifications and design phases. The software team regularly plays catch-up to the more mature project. However, this maturity is often limited to the subsystem level and does not apply to the system as a whole. The software team serves double duty as both software developers and system engineers [8]. In order to develop software requirements, system questions must be asked and answered. Often, these questions are first asked in the software requirements specification effort. These questions cannot always be answered immediately and, if the software team waits until the system is fully defined (all the questions are answered), schedules would never be met. As the system evolves, system requirements will change imposing new requirements on the software. However, even with a relatively stable system, the unstable process remains.

3.2 Documentation

Documentation alone is a weak communication medium [9]. CFG gathered and merged requirements from three NASA centers and four systems. Conflict among requirements is not uncommon in large systems [9] and the CFG was no exception. All PGSC commands are routed through the SFMDM. However, [5] stated that all SFMDM commands were to be 32 words in length while all PGSC commands were to be 30 words in length. This ambiguity was not uncovered until End-to-End test at JSC, very late in the project time

line. If more verbal communication had been possible earlier, this problem could have been caught while we were still developing at LaRC. The documentation could have been clarified and the problem solved much faster.

3.3 Detail Comprehension

It is impossible to see every requirement at first glance. Many of the details become known as we progress through the implementation [10]. Often one does not know the questions that need to be asked until faced with an implementation question. Even if we knew every requirement, experience shows that people are unable to fully comprehend the plethora of details that must be taken into account in order to design and build a correct system [10]. For this reason, as the picture of what is required of the software system becomes clearer, changes will occur.

3.4 User Requirements

In many cases, those who commission the building of a software system cannot specify exactly what they want nor are they able to communicate everything they know up front. There is a cultural gap between the developer and the customer [11]. The developer has the skills necessary to meet the customer requirements but does not have a complete understanding of the application while the customer has a complete view of the application but cannot know the capabilities of the software. This is often the case with mission GSE. The user specifies that a system is needed for commanding or to display science data. Often, little thought has been given to exactly what the user wants to see and how the user will interact with the system. Scientists and engineers are busy at the subsystem level while the software developers are working at the system level. Software prototyping has been proposed as a solution to this problem and the generation of a strawman concept is useful in minimizing the impact of changing user interface requirements. However, we found that there is a limit to how far ahead a system can be prototyped. No amount of prototyping can predict the wants and wishes of the users once the system is integrated, new system hot spots have been discovered and the community of users changes.

3.5 External Influences

Because software is intangible, it is constantly subject to pressures for change [12]. This can be due to “creeping elegance”, an altered interface, “planned” flexibility, or the need to fix a system problem. In any case, no traces are cut, no connectors demated, and, usually, flight configuration is not broken when a software change is made. Software can be viewed as infinitely malleable and an inexpensive fix since it can be changed “more easily” than hardware. To some degree, this is true, especially in the later phases of the project life cycle. When external interfaces change, the software is often expected to change in order to accommodate the new requirements. When a new capability of deemed necessary, software is often expected to provide this functionality.

4. Software Development Approaches

Three widely accepted approaches to software development are the waterfall life cycle model, the spiral life cycle model and prototyping. While we recognized the merit of these schemes, careful examination proved that no single one of them would meet our needs given the immature system requirements and the accelerated project schedule. We selected the most fitting aspects from these well-documented models to create an approach that suited our development environment.

4.1 Waterfall

The waterfall model consists of five basic stages (requirements analysis and definition, system and software design, implementation and unit testing, system testing, operations and maintenance) and assumes that one stage ends before the next stage begins [13]. In the typical flight software development environment, strict adherence to waterfall development methodologies is impossible [14]. The uncertainties of high-level requirements definition and significant hardware and spacecraft interface design changes well into the software development cycle would prevent ever evolving out of the software requirements phase. This was the case during the development of the Pallet Simulator Master AT. If we had waited until all requirements were firm before moving on to the design phase, we would have had nothing but requirements to show on the designated delivery date.

4.2 Spiral

The spiral model [15] is a risk-driven approach that emphasizes prototyping and incremental and iterative deliveries and allows a more realistic approach to requirements satisfaction than the traditional waterfall models [16]. Each iteration around the spiral passes through four phases (Planning, Risk Analysis, Engineering, Customer Evaluation) as progressively more complete versions of the software are built. It attempts to combine the best elements of the waterfall model and prototyping while recognizing and providing a mechanism for risk analysis. In that, the spiral model demands considerable risk assessment expertise and we simply did not have the time or the expertise to fully implement the spiral model development approach.

4.3 Prototyping

Software prototyping enables the developer to create a model of the software that must be built. It is especially helpful in the requirements specification phase [11]. Because we had a set of general objectives for the GSE software but little detailed information about input, processing, and output requirements, prototyping seemed the best modeling choice, especially for gaining insight into the user interface. However, we did not have the time to implement a pure prototyping life cycle. We could not afford to throw away precious development time along with the models themselves. However, prototyping would enable us to create a model of the software that, at a minimum, would simulate the user-machine interface. This “strawman” approach worked well. It defined what the user interface

should look like and we were able to extract other system requirements. In GSE, very little processing is performed that is not displayed in one form or another. Prototyping led to a top-down function design process that meshed well with the real-time context. The first "quick design" prototype was a throw-away. After a look and feel was agreed upon, we moved to a data driven approach.

4.4 Data Driven Approach

Many real-time system developers feel that present-day software engineering methods are insufficient to meet the needs of real-time programming [17]. Data flow-oriented techniques are generally seen as the best fit method for real-time system design. Functional decomposition makes it easy to cope with some peculiar aspects of real-time systems such as concurrency and timing requirements [18]. For non-embedded applications like the CFG, prototyping is helpful in designing the user interface from which requirements inevitably fall out. We found this to be true in the Pallet Simulator development but wanted a more modular design for the CFG. Due in part to the language selection (Ada), the design of the Master AT was inherently modular enough to allow for reuse of packages whose functionality would be required later (i.e. LITE command block construction) but we knew we could improve on that with the CFG. In general, functional oriented methods are stronger in the early stages of software design in going from requirements to top-level design elements. We carried over this philosophy to the CFG but used an Object Oriented Design (OOD) approach at the package level. Our high-level requirements were functional in nature and mapping the CFG requirements back to the mission requirements was simple. We defined objects to represent the interfaces and data definitions that required further clarification. This enabled us to isolate the code that we knew would change, as requirements were refined and system changes impacted current requirements. This helped in driving the changes into the data but was not sufficient. We did not want to perform a system rebuild every time a data definition changed. External data files could be used to provide dynamic data definition. The following narrates the design decisions and techniques used to minimize risk by pushing the system uncertainty into the data.

5. Design Techniques

The selected language for flight and ground software development was Ada. Ada facilitated the use of a number of design techniques such as information hiding, data abstraction and modularity. These techniques were successfully employed in the generation of several reusable packages for menuing, operating system isolation and data display.

5.1 Ada Language Choice

Ada was developed in the early 1980's as the mandated language of the Department of Defense (DOD) and as a potential solution to the rising cost of software development. NASA LaRC had not mandated the use of Ada in its flight projects but the organization responsible for the LITE software was in the process of establishing a center-wide

standard for software. The software engineer on LITE selected Ada as the language best suited for real-time system development at LaRC for both embedded and non-embedded environments. Ada was not specifically manufactured to support object oriented design. Nevertheless, some of its features like data and processing abstraction, generic types, packages (that support information hiding), are particularly useful in implementing object oriented design. The "package" is an important Ada concept. Inherent in the use of the Ada package is a certain degree of modularity. Even though OOD was not used in the development of the Master AT, all of the functions also needed by the CFG system could be ported directly over with only a few modifications. Ada was especially useful in the design phase of the CFG. The design was a team effort with individual responsibilities assigned to each member. During the design, specifications for all packages were created and compiled. These specifications imported other specifications so we were able to group blocks and subsystems before any processing code was created. Package bodies were created with procedure and function stubs. The coding phase simply filled in these functions and procedures one by one. Integration and test was simplified in that we already had a system in place. We needed only to verify functional correctness. Ada promotes software engineering better than any other modern language [19]. The package concept, strong typing and Ada compiler helped us enforce our software interfaces and imposed a modular design on both systems. Adherence to the Ada philosophy inherent in the language provided modularization, abstract data types and hierarchical structuring [12].

5.2 Modularity

Modularity has a direct positive impact on maintainability [18]. To this end, the design of the CFG system was modular in nature. At the top level, functional decomposition separated logical modules. Within each functional module, objects were separated though the use of packages using an OOD approach. A high-level architectural block diagram can be found in Figure 4. Due to immature requirements during the initial design phase, we often did not know the exact definition of that data that was to be passed from module to module. We knew the size of the data but the format was yet to be determined. To work around this problem, we were forced to override one of the features of Ada, strong typing. To simplify parameter passing to and from procedures, all parameters were passed as arrays of unsigned bytes. Unchecked Conversion allowed for the integration of many different data types used within individual packages. When the data definitions were defined, the existing procedures and functions were overloaded to handle both the primitive data (unsigned byte) and the strongly typed data type.

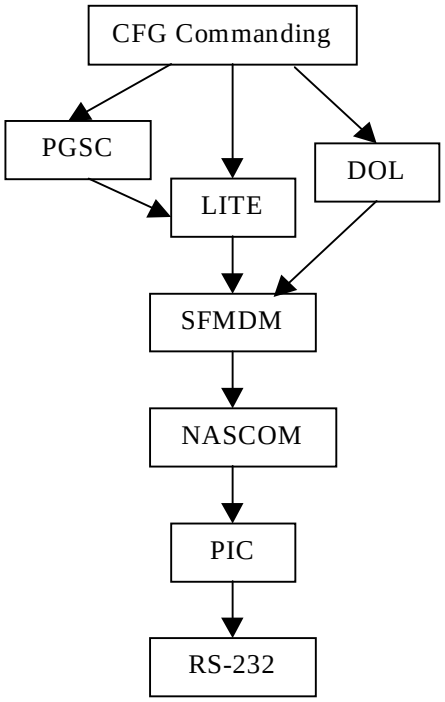

Figure 4: Partial CFG Architecture

5.3 Abstraction and Information Hiding

Information hiding focuses on what to hide; abstraction focuses on what to reveal. The principle of information hiding is that each module (or package) hides a design decision and implementation. If the design decision changes later, then no calling code should need to change. It is the other side of the coin from data abstraction [20]. Abstract data types should make it possible to change the data representation of objects while guaranteeing that all visible operations on the objects have the same properties. Procedural abstraction can be used to obtain much of the effect of abstract data types. A data structure can be defined and a set of procedures can be written to manipulate that data [21]. In Ada, data abstraction is often accomplished through the use of generic packages. In the CFG development, abstraction was achieved through procedures for ease of testing. Generic packages presented difficulties in the test and debug phase of software development. The inline debuggers could not see inside an Ada generic. Further, in Ada 83, generics were highly inefficient in memory usage and reclamation at execution time.

The Menu Package (Appendix 1) is one example of data abstraction and information hiding used in the software. It provides a text-based menuing facility that was developed for the user interfaces on both the Master AT and CFG platforms. Neither system had graphics displays requirements. Therefore, text-based menuing and data display was chosen for speed and executable size advantages. With the Menu Package, vertical menus were defined, displayed and read via the limited private type `Menu_Type` and the

procedures and functions provided for operations on a menu object. Menus are defined by procedure `Define_Menu` using upper left and lower right coordinates, color scheme, title, and border style. The parameters are read from an input file specified by the calling routine. This is an excellent example of information hiding. The programmer creates a text file that contains all the information required to define a menu and declares an object of type `Menu_Type` with no need to know exactly what this type looks like. Procedures and functions are provided for access information about the menu. `Get_Menu_Coordinates`, `Get_Menu_Color_Scheme`, `Get_Menu_Border_Style` and `Get_Menu_Title` are examples of the procedural data abstraction used in providing operations on the `Menu_Type`. `Read_Menu` returns the sequence number of the selected item in the menu specified. Other operations are `Append_Item`, `Insert_Item`, and `Delete_Item` that allow menu modification during program execution.

While the Menu Package was created specifically for menuing, we also used it during the prototyping phase of development. Menu parameters can be defined such that no highlighted selection bar is present and dummy data can quickly and easily be placed on the screen with calls to `Define_Menu` and `Display_Menu`. Data can then be moved around on the screen and colors changed with a simple modification to the input text files with no required recompilation, bind or link. This is discussed in more depth in Section 5.7.

5.4 Filename Manager Package

The Filename Manager Package allowed greater flexibility in naming external files such as data files, initialization files, and menu definition files. It allowed the code to use identifier strings, called "tags", to select files, instead of being limited to the file naming restrictions imposed by the host operating system. In addition, it also prevented literal file names from being "hard-wired" into the code. This allowed us to change file names, or use a substitute file with a different name in a test, without having to modify or re-compile the code. It also provided operating system independence.

An input file is created that contains a list of data pairings, the actual file name and its associated file tag. A pairing in an input file appears as

```
[CFG_Command_Logfile]
[C:\CFG\LOG\COMMAND.LOG]
```

where the file tag `CFG_Command_Logfile` is mapped to the actual file `COMMAND.LOG` in directory `C:\CFG\LOG\`.

The actual file names are assigned when the file is read in at list initialization. This allows complete code isolation from the operating system. It also allows for easy testing of new data files. The filename input file can be modified to associate a different file name with an existing file tag and the new file name will be used. No modifications are required to an existing working file so nothing is lost in testing the new data definition.
The following functions are available operations on the private type

`Filename_List_Type` that is a record that contains the filename list and the length of the list. Private type `File_Node_Type` defines the list contents as a record that holds the file tag, the actual filename to which the tag maps and a pointer to the next list element. The private type declarations are listed below. Filename Manager Package, in its entirety, can be found in Appendix 2.

```
TYPE Filename_List_Type IS
     RECORD
        Length      : NATURAL: = 0;
        Node_List   : File_Node_Access_Type;
     END RECORD;

TYPE File_Node_Type;
TYPE File_Node_Access_Type IS ACCESS File_Node_Type;
TYPE File_Node_Type IS
     RECORD
        File_Tag     : Variable_String_Package.VString;
        Filename     : Variable_String_Package.VString;
        Next    : File_Node_Access_Type;
     END RECORD;
```

The overloaded function `Get_Filename` searches the specified list for the file tag or tag position passed to it and returns the actual file name associated with that tag. A mapping is created with procedure `Generate_Filename_List` that requires a filename list and an actual filename, `From_File`, which can be passed in as a result of the function `Get_Filename`. File tags can be retrieved with function `Get_File_Tag` based on a file name or position and specified list. Careful file management allows the position returned to be used in combination with the Menu Package choice in `Read_Menu`. Pairings can be removed from the set by either file name or file tag with procedures `Delete_File_Name` and `Delete_File_Tag`, respectively. Pairings can be inserted via procedures `Append_To_Filename_List` or `Insert_Into_Filename_List`. Procedure `Insert_Into_Filename_List` inserts the new pairing in sorted order, assuming a sorted list, by either file name or file tag as specified in the parameter `Insert`. Sorted mappings are achieved through procedure `Sort_Filename_List` which sorts by either file tag or filename. Procedure `Save_Filename_List` allows the user to save the current contents of a filename list to an external file so that runtime changes to a list are not lost. If `Save_Filename_List` is not called at shutdown, the original file from which the list was created will remain intact.

5.5 External Initialization Files

A big challenge for the Master AT development was lack of command and data definition. The data size had been determined, as had the maximum command size. However, little consideration had been given to the command and data format. Additionally, the LITE subsystems were not yet completed, further complicating command definition. Even once the basic command format was defined, the command codes themselves were still changing. Each LITE serial command had a unique associated 8-bit identifier when

coupled with the 2-bit command subsystem destination. Three initialization files were created - one for each subsystem destination. At runtime, these files were read in and the command database was initialized from these files. As the subsystem command codes were defined, we needed only to edit the external text files in order to accommodate the system changes.

5.6. Constants Package

Figure 5 illustrates the evolution of the Constants Package concept used in both the Master AT and CFG systems. The following sections explain the refinement of this package that can be found in its entirety in Appendix 3.

5.6.1. Specification only. While developing the Pallet Simulator and CFG software, the team noticed that many constants and objects were needed in several packages resulting in multiple declarations. For example,

```
Byte_size : constant integer := 8;
```

appeared in almost every package body developed. If a requirements change effected any of these variables, the domino effect would have been significant. Our solution was the Constants Package that was used in both systems. A package Specification was created to isolate shared data definitions. Theoretically, this makes maintenance easier in that a change in one place will trickle down through the entire software system. However, there was one major drawback. While changes were isolated, a complete system rebuild was required whenever a change to the constants package was made. Many packages WITHed in the constants specification and had to be recompiled whenever the constants package specification was recompiled.

5.6.2. Specification and Body (Package) with function calls. In order to avoid a recompile of the entire system when a data definition changed, the existing constants package Specification was modified. Each object in the Specification was redefined as a function whose name was identical to the existing object name. No code modifications of existing software were required since the object to function translation was transparent to the calling code. The data values were defined as constant declarations in the package body. If a data definition changed, all that was required was a constants package body recompile which did not necessitate any further recompiles. All that was required was a new BIND and, as with the constants specification concept, the changes trickled down though the software. This was an improvement over the original specification only package but still did not isolate the risk to the data exclusively.

5.6.3. Package with initialization files. In order to eliminate recompiles entirely, the constants package body was further refined. At runtime, an initialization file was read in. All constant declarations were changed to object declarations. The objects declared were defined at runtime with the contents of the initialization file. With this modification, data definition changes could be affected with no compilation, no bind and no link of the system. The code was data independent and, as much as possible, data definition changes

were isolated to simple ASCII text files.

5.6.4. Package with initialization files and filename manager package. The final modification to the Constants Package was for OS isolation. The filename manager package was used in opening, reading and closing the initialization file. While this change did not increase data isolation, it was in keeping with the OS-independent design approach.

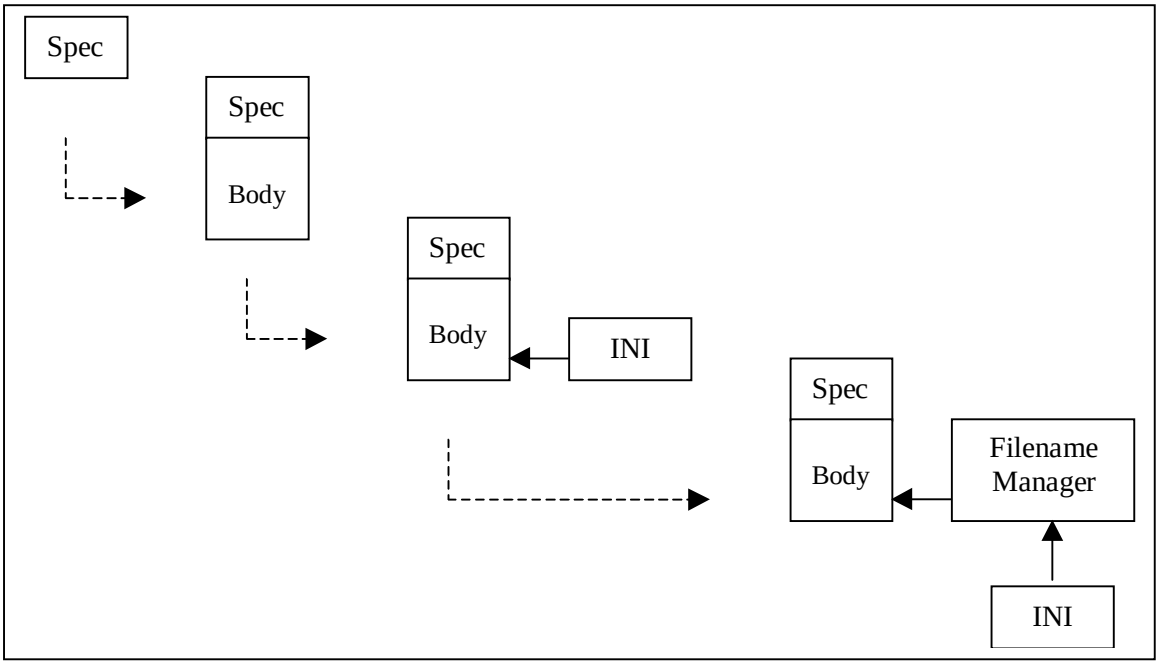

Figure 5 : Evolution of Constants Package

5.7. *Relative Positioning*

Relative referencing combines the data abstraction and information hiding illustrated by the Menu Package with the data-driven design approach of initialization files. The prototyping efforts accomplished with the Menu Package were so successful that we wanted to incorporate this idea into the delivered product. With the prototype, dummy data was used for display purposes. Static displays were sufficient for conveying the look and feel of the user interface. We needed a way to handle periodic, dynamic data for the working systems in both the test and mission environments and accomplished this through relative positioning of data. Menu files were used to define the placement, size and color of a data block. For example, a window to display command tracking is illustrated in Figure 6.

Commands sent from CFG:
Commands rejected by MCC:
SFMDM Input command counter:
SFMDM Command reject count:
Invalid Uplink Commands:

Figure 6: Command Count Data Window

This window can be created with a text file that contains the information in Figure 7.

```
[]        -- Menu Title
[0]       -- initial value of Current_Item
[2]       -- Upper Left Row
[2]       -- Upper Left Column
[8]       -- Lower Right Row
[30       -- Lower Right Column
[Blue_Background         -- Background Color
[Yellow_Foreground]      -- Foreground Color
[Yellow_Foreground]      -- Title Color
[Light_Cyan_Foreground]  -- Border Color
[Bright_White_Foreground]-- Highlight Color
[Single]                 -- Border Type

*************** Menu Items ****************

[0]
[Commands sent from CFG:]
[0]
[Commands rejected by MCC:]
[0]
[SFMDM Input Command Counter:]
[0]
[SFMDM Command Reject Count:]
[0]
[Invalid Uplink Commands:]
```

Figure 7: Command Count Window File

Defining and displaying the data blocks was simple. We needed a way to update the data that was equally as simple and flexible. Relative positioning was the solution to this. The Command Count Window was defined as a menu:

```
Command_Count_Window            : Menu_Package.Menu_Type;
```

Objects were declared to hold the row and column positions of the window.

```
Command_Count_Window_Top_Row,
Command_Count_Window_Bottom_Row : Screen_Package.Row_Type;
Command_Count_Window_Left_Column,
Command_Count_Window_Right_Column:Screen_Package.Column_Type;
```

At start-up, the menu is defined using the Filename Manager Package to avoid the use of OS-specific file names and reduce recompilations.

```
      Menu_Package.Define_Menu
         (The_Menu   => Command_Count_Window,
          Item_List  => Command_Count_Window_List,
          Filename   =>
               CFG_Filename_Package.Get_CFG_Filename
               (CFG_List => CFG_Filename_Package.MENU,
                File_Tag => Command_Count_Window_File_Tag));
```

The menus coordinates are assigned after the external file is read in and the coordinates are defined.

```
      Menu_Package.Get_Menu_Coordinates
          (Command_Count_Window,
           Command_Count_Window_Top_Row,
           Command_Count_Window_Left_Column,
           Command_Count_Window_Bottom_Row,
           Command_Count_Window_Right_Column);
```

At this point, the cursor positioning for data updates is entirely relatively to the coordinates read in from the external menu file.

```
    Screen_Package.Position_Cursor
      (Row     => Command_Count_Window_Top_Row + 1,
       Column  => Command_Count_Window_Right_Column - 5);
```

Changes to the menu file will be reflected in all data updates and data placement can be easily altered without any system changes.

6. Conclusions

LITE flew as the primary payload on STS-64 in September of 1994. It is the first lidar system to be operated in space and the first to use lasers in space for study of the Earth's atmospheric environment. It provided the most accurate measure of cloud top heights to date and provided data on the depth and structure of cirrus and other thin clouds. The success of LITE has provided new information on the distribution and characteristics of clouds to increase our understanding of their role in the global climate system. The LITE instrument CDR was held in April of 1989. A software team was formed in late 1990 to

develop systems to support integration and test, and mission. These real time systems were developed under a tight schedule and, despite the late start, with immature system requirements. Steps had to be taken to incorporate flexibility into the support software. We had no choice but to design for change.

One of the clearest lessons in this discussion about "design for change" is that one must anticipate changes before one begins the design [22]. System architecture changes in the later stages of a software project are dangerous, costly, and common. In order to avoid system architecture changes, the areas of the software most likely to change must be identified and consideration given to why they will change. This process illuminates how requirements will change and methods for accommodating these changes can be designed into the deliverable. Life cycle modeling attempts to capture this process but required unavailable resources such as personnel and time. The software team developed a data driven approach to development that incorporated aspects of several models. In both software systems developed for LITE, data was the most likely target for change. Information hiding, low intermodule coupling achieved through modular design and object-oriented development were key in limiting the impact of requirements changes. Additionally, we attempted to build in modifiability by pushing the elements we knew would change out of the code into external files and designing for dynamic data definitions.

Acknowledgments

The author gratefully acknowledges fellow CFG and Master AT software development team members Kerry M. Gough and Mark A. Parks. The author is also grateful to Michael L. Nelson whose careful reviews of this paper have improved its clarity and readability.

References

[1] David M. Winker, Richard H. Couch, and M. Patrick McCormick, “An Overview of LITE: NASA's Lidar In-space Technology Experiment”, *IEEE Proceedings*, Vol. 84, No. 2, February1996, pp. 164-180.

[2] B. Danette Allen, Kerry M. Gough, Mark A. Parks, “Mission Operations Command File Generator Software Requirements Specification for Lidar In-Space Technology Experiment (LITE)”, Version 2.0, NASA LaRC Document No. LITE-04-8-01-01, May 5, 1995.

[3] “POCC Capabilities Document”, NSTS-21063-POC-CAP, Rev. A, PCN-1, October 1991.

[4] “LITE Instrument Controller Software Requirements Specification”, D-IC-SRS, April 1993.

[5] “LITE-1 DDCS/SFMDM Software Requirements Specification”, SLMDH-0183, McDonnell Douglas Space Systems Company, Huntsville, February 1993.

[6] Marvin V. Zelkowitz, “Resource Utilization During Software Development” NASA Contractor Report, NASA-CR-191609, 1988.

[7] Edward H. Bersoff and Alan M. Davis, “Impacts of Life Cycle Models on Software”, *Communications of the ACM*, Vol. 34, No. 8, August 1991, pp. 104 – 117.

[8] *Findings of the Software Process Improvement Initiative*, NASA Langley Research Center, October 12, 1997. Available: <http://fmad-www.larc.nasa.gov/mdob/users/jctown/SPII/FindingsBfrg_4.html>

[9] Bill Curtis, Herb Krasner, and Neil Iscoe, “A Field Study of the Software Design Process for Large Systems”, *Communications of the ACM*, Vol. 31, No. 11, November 1988.

[10] David Lorge Parnas and Paul C. Clements, “A Rational Design Process and How to Fake It”, *IEEE Transactions on Software Engineering*, Vol. SE-12, No. 2, February 1986, pp. 251 – 257.

[11] John McDermid, Software Engineer’s Reference Book, Butterworth-Heinemann Ltd., 1991. p. 40/3.

[12] Fred P. Brooks, “No Silver Bullet: Essence and Accidents of Software Engineering”, *Computer*, Vol. 20 No. 4, 1987, pp. 10 - 20.

[13] W. W. Royce, “Managing the Development of Large Software Systems: concepts and techniques”, *Proceedings of IEEE WESTCON*, 1970, pp. 1 – 9.

[14] Anne B. Elson, “Hardware Impacts to Software Development Strategies – The History of the Mars Observer Payload Data Subsystem Embedded Real-time Software”, *Proceedings of the AIAA Computers in Aerospace VII Conference*, AIAA Paper 89-3078, October 1989.

[15] B. Boehm, “A Spiral Model for Software Development and Enhancement”, *Computer*, Vol. 21, No. 5, May 1988, pp. 61 – 72.

[16] Herbert P. Woodward, “Developing Better Software: A Five Year History of Software Engineering Advancement”, *NASA Goddard Space Flight Center Proceedings of the Sixteenth Annual Software Engineering Workshop*, December 1991, pp. 283 – 301.

[17] D. Harel, H. Lachover, A. Naamad, A. Pnueli, M. Politi, R. Sherman, and A. Shtul-Trauring, “STATEMATE: A Working Environment for the Development of Complex Reactive Systems”, *Proceedings of the Tenth International Conference on Software Engineering*, April 1988, pp. 396 – 406.

[18] P. Occelli, “Object versus Functional Oriented Design”, AGARD, Aerospace Software Engineering for Advanced Systems, May 1993.

[19] Herbert P. Woodward, “Developing Better Software: A Five Year History of Software Engineering Advancement”, *Proceedings of the Sixteenth Annual Software Engineering Workshop*, SEL-91-006, NASA Goddard Space Flight Center Software Engineering Laboratory, December 1991, pp. 283 – 301.

[20] David Alex Lamb, Software Engineering: Planning for Change, Prentice Hall, 1988, p. 48.

[21] Theodore A. Linden, “The Use of Abstract Data Types to Simplify Program Modifications”, Contribution of the National Bureau of Standards, 1984.

[22] David L. Parnas, “Designing Software for Ease of Extension and Contraction”, *Proceedings of the 3rd International Conference on Software Engineering*, May 10-12, 1978, pp. 264 – 277.

Appendix 1: Menu Package

```
-------------------------------------------------------------------------------
-- File Name:       MENU.ADS
-- Organization:    NASA - Langley Research Center (LaRC)
-- Project:         LIDAR In-Space Technology Experiment (LITE)
-------------------------------------------------------------------------------
--*****************************************************************************
-- Name/Number:
--    Menu_Package                                          (package spec)
--
-- Abstract:
--    This package provides provides a basic menuing facility. Vertical
--    menus can be defined, displayed,
--    This package relies on the device driver ANSI.SYS.
--
-- Acronyms/Abbreviations:
--
-- Dependencies:
--    Dynamic_String_Package
--    List_Package
--    Keyboard_Package
--    Screen_Package
--
-- Global Objects:
--    NONE
--
-- Exceptions:
--    NONE
--
-- Machine/Compiler Dependencies:
--   NONE
--
--*****************************************************************************

with Variable_String_Package;
with List_Package;
with Keyboard_Package;
with Screen_Package;

package Menu_Package is

   Subtype VString is Variable_String_Package.VString;

   type Menu_Item_Type is
      record
         Item_VString     : VString;
         Hot_Key_Position : Natural range 0 .. 100 := 0;
      end record;

-------------------------------------------------------------------------------
   Procedure Dispose_VString (Menu_Item : in out Menu_Item_Type);

   package Menu_List_Package is new List_Package (Item_Type => Menu_Item_Type,
                                                  Dispose   => Dispose_VString);
-------------------------------------------------------------------------------

   type Border_Type is (Single,Double,Double_Vertical_Border,Double_Horizontal_Border);

   type Color_Scheme_Type is
      record
         Background_Color : Screen_Package.Background_Color_Type;
         Foreground_Color : Screen_Package.Foreground_Color_Type;
```

```
         Title_Color      : Screen_Package.Foreground_Color_Type;
         Border_Color     : Screen_Package.Foreground_Color_Type;
         Highlight_Color  : Screen_Package.Foreground_Color_Type;
      end record;

   type Clear_Screen_Type is (No_Clear_Screen, Clear_Screen);

   Default_Color_Scheme : constant Color_Scheme_Type :=(Screen_Package.Blue_Background,
                                                        Screen_Package.White_Foreground,
                                                        Screen_Package.Yellow_Foreground,
                                                        Screen_Package.White_Foreground,
                                                        Screen_Package.Red_Foreground);

   Default_Menu_Upper_Left_Row     : constant Screen_Package.Row_Type    := 10;
   Default_Menu_Upper_Left_Column  : constant Screen_Package.Column_Type := 10;
   Default_Menu_Lower_Right_Row    : constant Screen_Package.Row_Type    := 20;
   Default_Menu_Lower_Right_Column : constant Screen_Package.Column_Type := 20;

   type Menu_Type is limited private;

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------

     -- This procedure will draw a rectangular border as specified by
     -- by the diagonal coordinates below.  The border is drawn in the
     -- current foreground color.  This procedure requires that the
     -- video system support the extended character set of the IBM PC.
     -- The title will be displayed in the foreground and background
     -- colors as specified when the procedure "Set_Title_Colors" is called.
   procedure Draw_Border (Upper_Left_Row     : in Screen_Package.Row_Type    :=
                                               Default_Menu_Upper_Left_Row;
                          Upper_Left_Column  : in Screen_Package.Column_Type :=
                                               Default_Menu_Upper_Left_Column;
                          Lower_Right_Row    : in Screen_Package.Row_Type    :=
                                               Default_Menu_Lower_Right_Row;
                          Lower_Right_Column : in Screen_Package.Column_Type :=
                                               Default_Menu_Lower_Right_Column;
                          Color_Scheme       : in Color_Scheme_Type          :=
                                               Default_Color_Scheme;
                          Title              : in String                 := "";
                          Border_Style       : in Border_Type            := Single;
                          Clear_Mode         : in Clear_Screen_Type      :=
                                               No_Clear_Screen);

   -- Define a Menu ------------------------------------------------------------

     -- This procedure defines a menu containing a list of items with the given
     -- upper left & lower right coordinates, color scheme, title, border style.
     -- If the Current_Item field is left to its default, you cannot execute a
     -- Read_Menu command without raising a constraint error!
   procedure Define_Menu (The_Menu           : in out Menu_Type;
                          Item_List          : in Menu_List_Package.List_Type;
                          Current_Item       : in Natural                    := 0;
                          Upper_Left_Row     : in Screen_Package.Row_Type    :=
                                               Default_Menu_Upper_Left_Row;
                          Upper_Left_Column  : in Screen_Package.Column_Type :=
                                               Default_Menu_Upper_Left_Column;
                          Lower_Right_Row    : in Screen_Package.Row_Type    :=
                                               Default_Menu_Lower_Right_Row;
                          Lower_Right_Column : in Screen_Package.Column_Type :=
                                               Default_Menu_Lower_Right_Column;
                          Color_Scheme   : in Color_Scheme_Type := Default_Color_Scheme;
                          Title          : in String            := "";
```

```
                        Border_Style   : in Border_Type       := Single);

-- Define a Menu ------------------------------------------------------------

  -- This procedure defines a menu containing a list of items with the given
  -- upper left & lower right coordinates, color scheme, title, border style.
  -- The parameters are read from the file specified by the user.
procedure Define_Menu (The_Menu  : in out Menu_Type;
                       Item_List : in out Menu_List_Package.List_Type;
                       Filename  : in String);


-- Change specific attributes of an existing menu --------------------------

  -- reposition where the menu is on the screen
procedure Set_Menu_Coordinates (The_Menu          : in out Menu_Type;
                                Upper_Left_Row    : in Screen_Package.Row_Type;
                                Upper_Left_Column : in Screen_Package.Column_Type;
                                Lower_Right_Row   : in Screen_Package.Row_Type;
                                Lower_Right_Column: in Screen_Package.Column_Type);

  -- changes the list of items which are available for The_Menu
procedure Set_Menu_Item_List (The_Menu  : in out Menu_Type;
                              Item_List : in Menu_List_Package.List_Type);

procedure Set_Color_Scheme_Type (The_Menu     : in out Menu_Type;
                                 Color_Scheme : in Color_Scheme_Type);

procedure Set_Menu_Title (The_Menu : in out Menu_Type;
                          Title    : in String);

procedure Set_Menu_Border_Style (The_Menu     : in out Menu_Type;
                                 Border_Style : in Border_Type);

procedure Set_Current_Item (The_Menu : in out Menu_Type;
                            New_Item : in Natural := 0);


-- Define a Color Scheme ----------------------------------------------------
function Define_Color_Scheme(Background_Color:in Screen_Package.Background_Color_Type;
                             Foreground_Color:in Screen_Package.Foreground_Color_Type;
                             Title_Color     :in Screen_Package.Foreground_Color_Type;
                             Border_Color    :in Screen_Package.Foreground_Color_Type;
                             Highlight_Color :in Screen_Package.Foreground_Color_Type)
                             return Color_Scheme_Type;


-- Request specific info about a menu ---------------------------------------

procedure Get_Menu_Coordinates (The_Menu           : in Menu_Type;
                                Upper_Left_Row     : out Screen_Package.Row_Type;
                                Upper_Left_Column  : out Screen_Package.Column_Type;
                                Lower_Right_Row    : out Screen_Package.Row_Type;
                                Lower_Right_Column : out Screen_Package.Column_Type);

Function Get_Menu_Color_Scheme (The_Menu : in Menu_Type) return Color_Scheme_Type;

Function Get_Menu_Border_Style (The_Menu : in Menu_Type) return Border_Type;

Function Get_Menu_Title (The_Menu : in Menu_Type) return VString;
```

```
-- Initialize and append Items to an Item List -----------------------------

procedure Initialize_Item_List (Item_List : in out Menu_List_Package.List_Type);

procedure Append_Item (The_Item              : in String;
                       With_Hot_Key_Position : in Natural := 0;
                       To_List               : in out Menu_List_Package.List_Type);

procedure Insert_Item( The_Item              : in String;
                       With_Hot_Key_Position : in Natural := 0;
                       At_Position           : in out Natural;
                       In_List               : in out Menu_List_Package.List_Type);

procedure Delete_Item( At_Position : in out Natural;
                       From_List   : in out Menu_List_Package.List_Type);


  -- returns the number of items associated with the specified menu
function Number_Of_Items( In_Menu : in Menu_Type) return Natural;

-- Display Menu -------------------------------------------------------------
procedure Display_Menu (The_Menu : in Menu_Type);


-- Hide Menu ----------------------------------------------------------------
procedure Hide_Menu (The_Menu   : in Menu_Type;
                     Fill_Color : in Screen_Package.Background_Color_Type);


-- Read Menu ----------------------------------------------------------------
procedure Read_Menu (The_Menu   : in out Menu_Type;
                     The_Choice : out Natural);


-- Read Menu ----------------------------------------------------------------
  -- This version of Read_Menu adds two important funtions: one, it allows the
  -- use of a predefined array of special keys as additional exits from the
  -- menu, and two, it allows the user to define a special case for the use
  -- of the escape key.
procedure Read_Menu (The_Menu           : in out Menu_Type;
                     Legal_Special_Keys : in Keyboard_Package.Special_Keys_Array_Type;
                     Special_Key_Hit    : out Boolean;
                     Entered_Special_Key: out Keyboard_Package.Special_Keys_Type;
                     Escape_Key_Hit     : out Boolean;
                     The_Choice         : out Natural);


-- Undefine Menu ------------------------------------------------------------
procedure Undefine_Menu (The_Menu : in out Menu_Type);

-- Undefine Menu ------------------------------------------------------------
   -- Remove menu definition from memory ------------------------------------
   -- This version allows user to also clear the menu_list contained in the
   -- user's application code with deallocation of VStrings.
procedure Undefine_Menu (The_Menu : in out Menu_Type;
                         The_List : in out Menu_List_Package.List_Type);
```

```
private

   type Menu_Structure is
      record
         Title              : VString;
         Item_List          : Menu_List_Package.List_Type;
         Item_List_Length   : Natural                    := 0;
         Current_Item       : Natural                    := 0;
         Highlight_Bar      : Natural                    := 0;
         First_Window_Item  : Natural                    := 0;
         Last_Window_Item   : Natural                    := 0;
         Upper_Left_Row     : Screen_Package.Row_Type    := Default_Menu_Upper_Left_Row;
         Upper_Left_Column  : Screen_Package.Column_Type :=
Default_Menu_Upper_Left_Column;
         Lower_Right_Row    : Screen_Package.Row_Type    := Default_Menu_Lower_Right_Row;
         Lower_Right_Column : Screen_Package.Column_Type :=
Default_Menu_Lower_Right_Column;
         Color_Scheme       : Color_Scheme_Type          := Default_Color_Scheme;
         Border_Style       : Border_Type                := Single;
      end record;

   type Menu_Type is access Menu_Structure;

end Menu_Package;
```

•

```
-------------------------------------------------------------------------------
-- File Name:       MENU.ADB
-- Organization:    NASA - Langley Research Center (LaRC)
-- Project:         LIDAR In-Space Technology Experiment (LITE)
-------------------------------------------------------------------------------
--*****************************************************************************
-- Name/Number:
--    Menu_Package                                          (package body)
--
-- Abstract:
--    This package provides provides a basic menuing facility. Vertical
--    menus can be defined, displayed, read, and removed from memory.
--
-- Acronyms/Abbreviations:
--
-- Dependencies:
--    DOS
--    Get_Token
--    Text_IO
--    Keyboard_Package
--    Screen_Package
--    Unsigned
--    Unchecked_Conversion   !!! not required !!!
--
-- Global Objects:
--    NONE
--
-- Exceptions:
--    NONE
--
-- Machine/Compiler Dependencies:
--    NONE
--
--*****************************************************************************


with Get_Token;
with Text_IO;
with Keyboard_Package;
with Screen_Package;
with Unsigned;
with Unchecked_Conversion;
with Unchecked_Deallocation;


package body Menu_Package is


   -- Functions for explicit type conversions.
   function To_Character is new Unchecked_Conversion (Source => Unsigned.Byte,
                                                      Target => Character);

   function Convert_Special_Key_To_Char is new Unchecked_Conversion
                           (Source => Keyboard_Package.Special_Keys_Type,
                            Target => Character);

     -- for undefining menus
   procedure Dispose_Of_Menu is new Unchecked_Deallocation( Menu_Structure, Menu_Type);

   Single_Upper_Left_Corner  : constant Character := To_Character( 218);
   Single_Upper_Right_Corner : constant Character := To_Character( 191);
   Single_Lower_Left_Corner  : constant Character := To_Character( 192);
   Single_Lower_Right_Corner : constant Character := To_Character( 217);
```

```
   Single_Vertical   : constant Character := To_Character( 179);
   Single_Horizontal : constant Character := To_Character( 196);

   Double_Upper_Left_Corner  : constant Character := To_Character( 201);
   Double_Upper_Right_Corner : constant Character := To_Character( 187);
   Double_Lower_Left_Corner  : constant Character := To_Character( 200);
   Double_Lower_Right_Corner : constant Character := To_Character( 188);

   Double_Vertical_Upper_Left_Corner  : constant Character := To_Character( 214);
   Double_Vertical_Upper_Right_Corner : constant Character := To_Character( 183);
   Double_Vertical_Lower_Left_Corner  : constant Character := To_Character( 211);
   Double_Vertical_Lower_Right_Corner : constant Character := To_Character( 189);

   Double_Horizontal_Upper_Left_Corner  : constant Character := To_Character( 213);
   Double_Horizontal_Upper_Right_Corner : constant Character := To_Character( 184);
   Double_Horizontal_Lower_Left_Corner  : constant Character := To_Character( 212);
   Double_Horizontal_Lower_Right_Corner : constant Character := To_Character( 190);

   Double_Vertical   : constant Character := To_Character( 186);
   Double_Horizontal : constant Character := To_Character( 205);

   Highlight_Row_Cursor_Position    : Screen_Package.Row_Type;
   Highlight_Column_Cursor_Position : Screen_Package.Column_Type;


   -- Arithmetic/Logical function renamings for Row_Types and Column_Types defined in
   -- Screen_Package.

function "+" (Row1, Row2 : in Screen_Package.Row_Type) return Screen_Package.Row_Type
renames Screen_Package."+";

function "-" (Row1, Row2 : in Screen_Package.Row_Type) return Screen_Package.Row_Type
renames Screen_Package."-";

function ">=" (Row1, Row2 : in Screen_Package.Row_Type) return Boolean renames
Screen_Package.">=";

function "+" (Column1,Column2:in Screen_Package.Column_Type) return
            Screen_Package.Column_Type renames Screen_Package."+";

function "-" (Column1, Column2 : in Screen_Package.Column_Type) return
              Screen_Package.Column_Type renames Screen_Package."-";

function ">=" (Column1, Column2 : in Screen_Package.Column_Type) return Boolean renames
            Screen_Package.">=";

function "/" (Column1, Column2 : in Screen_Package.Column_Type) return
Screen_Package.Column_Type renames Screen_Package."/";


   ------------------------------------------------------------------------------

   ------------------------------------------------------------------------------
Procedure Dispose_VString (Menu_Item : in out Menu_Item_Type) IS


   Begin
      Variable_String_Package.Destroy(Menu_Item.Item_VString);
   end Dispose_Vstring;

------------------------------------------------------------------------------
```

```
   ------------------------------------------------------------------------------

   ------------------------------------------------------------------------------

   function MIN (Num1, Num2 : in Natural) return Natural is
   begin

      if Num1 <= Num2 then
         return Num1;
      else
         return Num2;
      end if;

   end MIN;

   -- ==========================================================================

   function MAX (Num1, Num2 : in Natural) return Natural is
   begin

      if Num1 >= Num2 then
         return Num1;
      else
         return Num2;
      end if;

   end MAX;

   -- ==========================================================================

   function Menu_Window_Length (The_Menu : in Menu_Type) return Natural is
   begin

      if The_Menu.Upper_Left_Row >= The_Menu.Lower_Right_Row then
         return 0;
      else
         return Natural (The_Menu.Lower_Right_Row - The_Menu.Upper_Left_Row - 1);
      end if;

   end Menu_Window_Length;

   -- ==========================================================================

   function Menu_Window_Width (The_Menu : in Menu_Type) return Natural is
   begin

      if The_Menu.Upper_Left_Column >= The_Menu.Lower_Right_Column then
         return 0;
      else
         return Natural (The_Menu.Lower_Right_Column - The_Menu.Upper_Left_Column - 1);
      end if;

   end Menu_Window_Width;

   ------------------------------------------------------------------------------
-------------------------------------------------------------------------------
   procedure Internal_Draw_Border (Upper_Left_Row     : in Screen_Package.Row_Type    :=
Default_Menu_Upper_Left_Row;
                           Upper_Left_Column  : in Screen_Package.Column_Type :=
Default_Menu_Upper_Left_Column;
                           Lower_Right_Row    : in Screen_Package.Row_Type    :=
                                                Default_Menu_Lower_Right_Row;
```

```
                        Lower_Right_Column : in Screen_Package.Column_Type :=
                                             Default_Menu_Lower_Right_Column;
                        Color_Scheme       : in Color_Scheme_Type          :=
                                             Default_Color_Scheme;
                        Title              : in String                     := "";
                        Border_Style       : in Border_Type                := Single;
                        Clear_Mode         : in Clear_Screen_Type          :=
                                             No_Clear_Screen) is

   Blank_Line : constant String( Positive (Upper_Left_Column + 1) .. Positive
                         (Lower_Right_Column - 1)) := (others => ' ');

   Horiz_or_Vert_Bar : Character;    -- jnr, 9/13/91

   Top_Row             : Screen_Package.Row_Type;
   Bottom_Row          : Screen_Package.Row_Type;
   Left_Column         : Screen_Package.Column_Type;
   Right_Column        : Screen_Package.Column_Type;
   Available_Positions : Screen_Package.Column_Type;
   Start_Offset        : Screen_Package.Column_Type;

   Saved_Foreground_Color : Screen_Package.Foreground_Color_Type;
   Saved_Background_Color : Screen_Package.Background_Color_Type;

begin -- Internal_Draw_Border

     -- Save the cursor position and colors so that they may be restored later.
   Screen_Package.Save_Cursor_Position;
   Saved_Foreground_Color := Screen_Package.Present_Foreground_Color;
   Saved_Background_Color := Screen_Package.Present_Background_Color;
     -- Set the border color.
   Screen_Package.Set_Foreground_Color (Color_Scheme.Border_Color);
   Screen_Package.Set_Background_Color (Color_Scheme.Background_Color);
     -- Position the cursor to the upper left corner and draw the
     -- appropriate upper left corner symbol.
   Screen_Package.Position_Cursor (Upper_Left_Row, Upper_Left_Column);

     -- set horizontal bar character
   case Border_Style is
      when Single =>
         Screen_Package.Put (Single_Upper_Left_Corner);
         Horiz_or_Vert_Bar := Single_Horizontal;
      when Double =>
         Screen_Package.Put (Double_Upper_Left_Corner);
         Horiz_or_Vert_Bar := Double_Horizontal;
      when Double_Vertical_Border =>
         Screen_Package.Put (Double_Vertical_Upper_Left_Corner);
         Horiz_or_Vert_Bar := Single_Horizontal;
      when Double_Horizontal_Border =>
         Screen_Package.Put (Double_Horizontal_Upper_Left_Corner);
         Horiz_or_Vert_Bar := Double_Horizontal;
   end case;


     -- Print the horizontal symbols for the top part of the border.
   Left_Column  := Upper_Left_Column + 1;
   Right_Column := Lower_Right_Column - 1;

   for Horizontal_Bar in Left_Column .. Right_Column loop
      Screen_Package.Put (Horiz_or_Vert_Bar);
   end loop;

     -- Draw the appropriate upper right corner symbol.
```

```
case Border_Style is
   when Single =>
      Screen_Package.Put (Single_Upper_Right_Corner);
   when Double =>
      Screen_Package.Put (Double_Upper_Right_Corner);
   when Double_Vertical_Border =>
      Screen_Package.Put (Double_Vertical_Upper_Right_Corner);
   when Double_Horizontal_Border =>
      Screen_Package.Put (Double_Horizontal_Upper_Right_Corner);
end case;

  -- If the user requests a title for this border, then determine how to
  -- center the title in the border.  If the title is longer than than the
  -- border, then the title will be truncated.  If the user allows the default
  -- null "" as the title, then no title will be printed.
if Title'Length > 0 then
     -- Set up the title foreground color.
   Screen_Package.Set_Foreground_Color (Color_Scheme.Title_Color);
   Available_Positions := Lower_Right_Column - Upper_Left_Column - 1;

   if Available_Positions >= Screen_Package.Column_Type( Title'Length) then
      Start_Offset := (Available_Positions - Screen_Package.Column_Type
                      (Title'Length)) / 2 + 1;
      Screen_Package.Position_Cursor
          (Upper_Left_Row,Upper_Left_Column+Start_Offset);
      Screen_Package.Put( Title);
   else
      Screen_Package.Position_Cursor (Upper_Left_Row, Upper_Left_Column + 1);
      Screen_Package.Put( Title( 1 .. Natural (Available_Positions)));
   end if;

end if;

  -- Restore the border foreground color.
Screen_Package.Set_Foreground_Color (Color_Scheme.Border_Color);
  -- Position the cursor to the lower left side of the menu so that the
  -- lower left corner symbol may be printed.
Screen_Package.Position_Cursor (Lower_Right_Row, Upper_Left_Column);

case Border_Style is
   when Single =>
      Screen_Package.Put (Single_Lower_Left_Corner);
   when Double =>
      Screen_Package.Put (Double_Lower_Left_Corner);
   when Double_Vertical_Border =>
      Screen_Package.Put (Double_Vertical_Lower_Left_Corner);
   when Double_Horizontal_Border =>
      Screen_Package.Put (Double_Horizontal_Lower_Left_Corner);
end case;



  -- Print the horizontal symbols for the bottom part of the border.
for Horizontal_Bar in Left_Column .. Right_Column loop
   Screen_Package.Put (Horiz_or_Vert_Bar);
end loop;

  -- Print the lower right corner symbol for the menu.
  -- set vertical bar character
case Border_Style is
   when Single =>
      Screen_Package.Put (Single_Lower_Right_Corner);
      Horiz_or_Vert_Bar := Single_Vertical;
```

```
      when Double =>
         Screen_Package.Put (Double_Lower_Right_Corner);
         Horiz_or_Vert_Bar := Double_Vertical;
      when Double_Vertical_Border =>
         Screen_Package.Put (Double_Vertical_Lower_Right_Corner);
         Horiz_or_Vert_Bar := Double_Vertical;
      when Double_Horizontal_Border =>
         Screen_Package.Put (Double_Horizontal_Lower_Right_Corner);
         Horiz_or_Vert_Bar := Single_Vertical;
   end case;

     -- Print both the left and right vertical sides of the menu bar.
   Top_Row    := Upper_Left_Row + 1;
   Bottom_Row := Lower_Right_Row - 1;

   for Vertical_Bar in Top_Row .. Bottom_Row loop

      Screen_Package.Position_Cursor (Vertical_Bar, Upper_Left_Column);
      Screen_Package.Put (Horiz_or_Vert_Bar);
      if Clear_Mode = Clear_Screen then
         Screen_Package.Put( Blank_Line);
      end if;
      Screen_Package.Position_Cursor (Vertical_Bar, Lower_Right_Column);
      Screen_Package.Put (Horiz_or_Vert_Bar);

   end loop;

     -- Restore the saved cursor position.
   Screen_Package.Restore_Cursor_Position;
     -- Restore the original foreground and background colors.
   Screen_Package.Set_Foreground_Color (Saved_Foreground_Color);
   Screen_Package.Set_Background_Color (Saved_Background_Color);

end Internal_Draw_Border;

-------------------------------------------------------------------------------
-------------------------------------------------------------------------------

procedure Draw_Border (Upper_Left_Row     : in Screen_Package.Row_Type    :=
                                            Default_Menu_Upper_Left_Row;
                       Upper_Left_Column  : in Screen_Package.Column_Type :=
                                            Default_Menu_Upper_Left_Column;
                       Lower_Right_Row    : in Screen_Package.Row_Type    :=
                                            Default_Menu_Lower_Right_Row;
                       Lower_Right_Column : in Screen_Package.Column_Type :=
                                            Default_Menu_Lower_Right_Column;
                       Color_Scheme : in Color_Scheme_Type:= Default_Color_Scheme;
                       Title        : in String                 := "";
                       Border_Style : in Border_Type            := Single;
                       Clear_Mode   : in Clear_Screen_Type      := No_Clear_Screen) is



begin -- Draw_Border

     -- Save the cursor position and colors so that they may be restored later.
   Screen_Package.Get_Screen;

   Internal_Draw_Border (Upper_Left_Row, Upper_Left_Column, Lower_Right_Row,
                         Lower_Right_Column,
                 Color_Scheme, Title, Border_Style,Clear_Mode);

   Screen_Package.Release_Screen;
```

```
end Draw_Border;

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------

  -- The following adjusts the value of The_Menu.Current_Item if it is
  -- too large, and also assigns values to the following fields in The_Menu:
  --    First_Window_Item
  --    Last_Window_Item
procedure Determine_Items_In_Window (The_Menu : in out Menu_Type) is

   Temp_Window_Length : Natural := 0;

begin

     -- make sure Current_Item is not beyond the end of the list
   if The_Menu.Current_Item > The_Menu.Item_List_Length then
      The_Menu.Current_Item := The_Menu.Item_List_Length;
   end if;

   Temp_Window_Length := Menu_Window_Length (The_Menu);

     -- if there are items and space to display them all, then do so.
   if The_Menu.Item_List_Length in 1 .. Temp_Window_Length - 1 then
      The_Menu.First_Window_Item := 1;
      The_Menu.Last_Window_Item  := The_Menu.Item_List_Length;

      -- there are items but only space to display some of them.
   elsif (The_Menu.Item_List_Length > 0) and (Temp_Window_Length > 0) then
        -- Make the Current_Item the top item in the window. If Current_Item = 0
        -- the top item will be item #1.
      The_Menu.First_Window_Item := MAX (1, The_Menu.Current_Item);
        -- Determine last item that will fit in the window, but don't go
        -- beyond the end of the Item_List.
      The_Menu.Last_Window_Item := MIN (The_Menu.Item_List_Length,
                                  The_Menu.First_Window_Item+Temp_Window_Length-1);
        -- If we are at the bottom of the list and the window is not full,
        -- move top of window above the Current_Item.
      The_Menu.First_Window_Item := The_Menu.Last_Window_Item+1-Temp_Window_Length;

   else     -- no items or no space.
      The_Menu.First_Window_Item := 0;
      The_Menu.Last_Window_Item  := 0;
   end if;

end Determine_Items_In_Window;

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------

procedure Define_Menu (The_Menu           : in out Menu_Type;
                       Item_List          : in Menu_List_Package.List_Type;
                       Current_Item       : in Natural                     := 0;
                       Upper_Left_Row     : in Screen_Package.Row_Type    :=
                                            Default_Menu_Upper_Left_Row;
                       Upper_Left_Column  : in Screen_Package.Column_Type :=
                                            Default_Menu_Upper_Left_Column;
                       Lower_Right_Row    : in Screen_Package.Row_Type    :=
                                            Default_Menu_Lower_Right_Row;
                       Lower_Right_Column : in Screen_Package.Column_Type :=
                                            Default_Menu_Lower_Right_Column;
```

```
                         Color_Scheme       : in Color_Scheme_Type        :=
                                              Default_Color_Scheme;
                         Title              : in String                   := "";
                         Border_Style       : in Border_Type              := Single) is

      VTitle : VString;

   begin

        -- Allocate storage for the new menu, if required
      Undefine_Menu (The_Menu);
      The_Menu := new Menu_Structure;

        -- Convert title to a "Variable String"
      Variable_String_Package.Make_VString (The_String => Title, New_VString => VTitle);
        -- Define the various fields in the menu
      The_Menu.Current_Item       := Current_Item;
      The_Menu.Upper_Left_Row     := Upper_Left_Row;
      The_Menu.Upper_Left_Column  := Upper_Left_Column;
      The_Menu.Lower_Right_Row    := Lower_Right_Row;
      The_Menu.Lower_Right_Column := Lower_Right_Column;
      The_Menu.Color_Scheme       := Color_Scheme;
      The_Menu.Title              := VTitle;
      The_Menu.Border_Style       := Border_Style;
      Set_Menu_Item_List (The_Menu => The_Menu, Item_List => Item_List);
        -- Determine which items to display, based on Current_Item.
      Determine_Items_In_Window (The_Menu => The_Menu);

   end Define_Menu;

   ------------------------------------------------------------------------------
   ------------------------------------------------------------------------------

   procedure Define_Menu (The_Menu  : in out Menu_Type;
                          Item_List : in out Menu_List_Package.List_Type;
                          Filename  : in String) is

      Length           : Natural;
      Hot_Key_Position : Natural;
      Token            : String (1 .. 80);
      Menu_File        : Text_Io.File_Type;
      VTitle           : VString;

   begin -- Define_Menu

      Text_Io.Open (File => Menu_File, Mode => Text_Io.In_File, Name => Filename);

        -- Allocate storage for the new menu, if needed
      Undefine_Menu (The_Menu);
      The_Menu := new Menu_Structure;


      Get_Token (Menu_File, Token, Length);
        -- Convert title to a "Dynamic String"

Variable_String_Package.Make_VString(The_String=>Token(1..Length),New_VString=>VTitle);
      The_Menu.Title := VTitle;
        -- Define the various fields in the menu
      Get_Token (Menu_File, Token, Length);
      The_Menu.Current_Item := Natural'VALUE (Token (1 .. Length));
      Get_Token (Menu_File, Token, Length);
      The_Menu.Upper_Left_Row := Screen_Package.Row_Type'VALUE (Token (1 .. Length));
      Get_Token (Menu_File, Token, Length);
```

```
   The_Menu.Upper_Left_Column := Screen_Package.Column_Type'VALUE (Token (1..Length));
   Get_Token (Menu_File, Token, Length);
   The_Menu.Lower_Right_Row := Screen_Package.Row_Type'VALUE (Token (1 .. Length));
   Get_Token (Menu_File, Token, Length);
   The_Menu.Lower_Right_Column := Screen_Package.Column_Type'VALUE(Token (1..Length));
     -- Set the color scheme
   Get_Token (Menu_File, Token, Length);
   The_Menu.Color_Scheme.Background_Color :=Screen_Package.Background_Color_Type'VALUE
                                            (Token (1 .. Length));
   Get_Token (Menu_File, Token, Length);
   The_Menu.Color_Scheme.Foreground_Color:= Screen_Package.Foreground_Color_Type'VALUE
                                            (Token (1 .. Length));
   Get_Token (Menu_File, Token, Length);
   The_Menu.Color_Scheme.Title_Color:= Screen_Package.Foreground_Color_Type'VALUE
                                            (Token (1 .. Length));
   Get_Token (Menu_File, Token, Length);
   The_Menu.Color_Scheme.Border_Color := Screen_Package.Foreground_Color_Type'VALUE
                                            (Token (1 .. Length));
   Get_Token (Menu_File, Token, Length);
   The_Menu.Color_Scheme.Highlight_Color := Screen_Package.Foreground_Color_Type'VALUE
                                           (Token (1 .. Length));
   Get_Token (Menu_File, Token, Length);
   The_Menu.Border_Style := Border_Type'VALUE (Token (1 .. Length));

   Initialize_Item_List( Item_List);

     -- read the items into list
   while not Text_IO.End_of_File (Menu_File) loop
      Get_Token (Menu_File, Token, Length);
      Hot_Key_Position := Natural'VALUE (Token (1 .. Length));
      Get_Token (Menu_File, Token, Length);
      Append_Item (Token (1 .. Length), Hot_Key_Position, Item_List);
   end loop;

   Set_Menu_Item_List (The_Menu, Item_List);

     -- Determine which items to display, based on Current_Item.
   Determine_Items_In_Window (The_Menu => The_Menu);
   Text_IO.Close (Menu_File);


exception
   when Text_Io.Name_Error =>
      Screen_Package.Put ("Menu file => " & Filename & " not found");
      raise;
   when Constraint_Error =>
      Text_IO.Close (Menu_File);
      Screen_Package.Put ("Invalid field value in Menu Initialization File");
      raise;
   when others =>
      Text_IO.Close (Menu_File);
      Screen_Package.Put ("Unanticipated error in subprogram Define Menu");
      raise;

end Define_Menu;

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------

procedure Set_Menu_Coordinates (The_Menu           : in out Menu_Type;
                                Upper_Left_Row     : in Screen_Package.Row_Type;
                                Upper_Left_Column  : in Screen_Package.Column_Type;
                                Lower_Right_Row    : in Screen_Package.Row_Type;
```

```
                                 Lower_Right_Column : in Screen_Package.Column_Type) is
begin

   The_Menu.Upper_Left_Row     := Upper_Left_Row;
   The_Menu.Upper_Left_Column  := Upper_Left_Column;
   The_Menu.Lower_Right_Row    := Lower_Right_Row;
   The_Menu.Lower_Right_Column := Lower_Right_Column;
     -- Changing coordinates may change window size
   Determine_Items_In_Window (The_Menu => The_Menu);

end Set_Menu_Coordinates;

-- ==========================================================================

procedure Set_Menu_Item_List (The_Menu  : in out Menu_Type;
                              Item_List : in Menu_List_Package.List_Type) is
begin

   The_Menu.Item_List := Item_List;
   The_Menu.Item_List_Length :=
                 Menu_List_Package.Length_Of(The_List=>The_Menu.Item_List);
     -- Changing Item_List may affect number of items available for display
   Determine_Items_In_Window (The_Menu => The_Menu);

end Set_Menu_Item_List;

-- ==========================================================================

procedure Set_Color_Scheme_Type (The_Menu     : in out Menu_Type;
                                 Color_Scheme : in Color_Scheme_Type) is
begin

   The_Menu.Color_Scheme := Color_Scheme;

end Set_Color_Scheme_Type;

-- ==========================================================================

procedure Set_Menu_Title (The_Menu : in out Menu_Type;
                          Title    : in String) is
   VTitle : VString;

begin

   -- Convert title to a "Variable String"
   Variable_String_Package.Make_VString (The_String => Title, New_VString => VTitle);
   The_Menu.Title := VTitle;

end Set_Menu_Title;

-------------------------------------------------------------------------------
-------------------------------------------------------------------------------

procedure Set_Menu_Border_Style (The_Menu     : in out Menu_Type;
                                 Border_Style : in Border_Type) is
begin

   The_Menu.Border_Style := Border_Style;

end Set_Menu_Border_Style;

-- ==========================================================================
```

```
procedure Set_Current_Item (The_Menu : in out Menu_Type;
                            New_Item : in Natural := 0) is
begin

   The_Menu.Current_Item := New_Item;

end Set_Current_Item;

-- ==========================================================================

function Define_Color_Scheme(Background_Color:in Screen_Package.Background_Color_Type;
                             Foreground_Color:in Screen_Package.Foreground_Color_Type;
                             Title_Color     :in Screen_Package.Foreground_Color_Type;
                             Border_Color    :in Screen_Package.Foreground_Color_Type;
                             Highlight_Color :in Screen_Package.Foreground_Color_Type)
                             return Color_Scheme_Type is
begin

   return (Background_Color,Foreground_Color,Title_Color,
           Border_Color,Highlight_Color);

end Define_Color_Scheme;

-- ==========================================================================

procedure Get_Menu_Coordinates(The_Menu           : in Menu_Type;
                               Upper_Left_Row     : out Screen_Package.Row_Type;
                               Upper_Left_Column  : out Screen_Package.Column_Type;
                               Lower_Right_Row    : out Screen_Package.Row_Type;
                               Lower_Right_Column : out Screen_Package.Column_Type) is
begin

   Upper_Left_Row     := The_Menu.Upper_Left_Row;
   Upper_Left_Column  := The_Menu.Upper_Left_Column;
   Lower_Right_Row    := The_Menu.Lower_Right_Row;
   Lower_Right_Column := The_Menu.Lower_Right_Column;

end Get_Menu_Coordinates;

-------------------------------------------------------------------------------
-------------------------------------------------------------------------------

function Get_Menu_Color_Scheme (The_Menu : in Menu_Type) return Color_Scheme_Type is
begin

   return The_Menu.Color_Scheme;

end Get_Menu_Color_Scheme;

-- ==========================================================================

function Get_Menu_Border_Style (The_Menu : in Menu_Type) return Border_Type is
begin

   return The_Menu.Border_Style;

end Get_Menu_Border_Style;

-- ==========================================================================

function Get_Menu_Title (The_Menu : in Menu_Type) return VString is
begin
```

```
   return The_Menu.Title;

end Get_Menu_Title;

-- ==========================================================================

procedure Initialize_Item_List (Item_List : in out Menu_List_Package.List_Type) is
begin

   Menu_List_Package.Clear (The_List => Item_List);

end Initialize_Item_List;

-------------------------------------------------------------------------------
-------------------------------------------------------------------------------

procedure Append_Item (The_Item              : in String;
                       With_Hot_Key_Position : in Natural := 0;
                       To_List               : in out Menu_List_Package.List_Type) is

   List_Position : Natural := Menu_List_Package.Length_Of( The_List => To_List) + 1;
   Temp_VString  : VString;
   Temp_Item     : Menu_Item_Type;

begin

     -- Convert the Standard.String The_Item to a variable string prior to assignment
   Variable_String_Package.Make_Vstring
                              (The_String=>The_Item,New_VString=>Temp_VString);
   Temp_Item.Item_VString := Temp_VString;

     -- Make sure position of hot key is valid.
   if With_Hot_Key_Position>Variable_String_Package.Length_Of(Temp_Item.Item_VString)
   then
      Temp_Item.Hot_Key_Position := 0;
   else
      Temp_Item.Hot_Key_Position := With_Hot_Key_Position;
   end if;

   Menu_List_Package.Insert_Item (The_Item => Temp_Item, In_List => To_List,
                                  At_Position => List_Position);

end Append_Item;

-- ==========================================================================

  -- Return the item following the Current_Item from the menu Item_List.
  -- Wrap around to the first if Current_Item is at the end of the list.
function Next_Item (Of_Menu : in Menu_Type) return Natural is
begin

   if Of_Menu.Current_Item = 0 then
      return 0;
   elsif Of_Menu.Current_Item < Of_Menu.Item_List_Length then
      return Of_Menu.Current_Item + 1;
   else
      return 1;
   end if;

end Next_Item;

-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
```

```
procedure Insert_Item( The_Item              : in String;
                       With_Hot_Key_Position : in Natural := 0;
                       At_Position           : in out Natural;
                       In_List               : in out Menu_List_Package.List_Type) is

   Temp_VString  : VString;
   Temp_Item     : Menu_Item_Type;

begin

     -- Convert the Standard.String The_Item to a variable string prior to assignment
   Variable_String_Package.Make_VString(
                           The_String=>The_Item,New_VString=>Temp_VString);
   Temp_Item.Item_VString := Temp_VString;

     -- Make sure position of hot key is valid.
   if With_Hot_Key_Position>Variable_String_Package.Length_Of(Temp_Item.Item_VString)
   then
      Temp_Item.Hot_Key_Position := 0;
   else
      Temp_Item.Hot_Key_Position := With_Hot_Key_Position;
   end if;

   Menu_List_Package.Insert_Item( The_Item=>Temp_Item, In_List=>In_List, At_Position
                                  => At_Position);

end Insert_Item;

-- ==========================================================================

procedure Delete_Item( At_Position : in out Natural;
                       From_List   : in out Menu_List_Package.List_Type) is
begin

   Menu_List_Package.Delete_Item( From_List => From_List, At_Position => At_Position);

end Delete_Item;

-- ==========================================================================

function Number_Of_Items( In_Menu : in Menu_Type) return Natural is
begin

   return In_Menu.Item_List_Length;

end Number_Of_Items;

-------------------------------------------------------------------------------
-------------------------------------------------------------------------------

  -- Return the item preceding the Current_Item from the menu Item_List.
  -- Wrap around to the last if Current_Item is the first item in the list.
function Prev_Item (Of_Menu : in Menu_Type) return Natural is
begin

   if Of_Menu.Current_Item = 0 then
      return 0;
   elsif Of_Menu.Current_Item = 1 then
      return Of_Menu.Item_List_Length;
   else
      return Of_Menu.Current_Item - 1;
   end if;
```

```
end Prev_Item;

-- ==========================================================================

  -- Returns true if The_Item_Number is within the menu's display window bounds
function Is_In_Window (The_Item_Number : in Natural;
                       Of_Menu         : in Menu_Type) return boolean is
begin

   if The_Item_Number = 0 then
      return false;
   --elsif The_Item_Number >= Of_Menu.First_Window_Item and The_Item_Number <=
   --Of_Menu.Last_Window_Item then
   elsif The_Item_Number in Of_Menu.First_Window_Item .. Of_Menu.Last_Window_Item then
      return true;
   else
      return false;
   end if;

end Is_In_Window;

-------------------------------------------------------------------------------
-------------------------------------------------------------------------------

  -- Displays a menu item in the given colors bounded by the given maximum width.
  -- Display occurs at the current screen cursor position.
procedure Display_Item (The_Item         : in Menu_Item_Type;
                        Background_Color : in Screen_Package.Background_Color_Type;
                        Foreground_Color : in Screen_Package.Foreground_Color_Type;
                        Highlight_Color  : in Screen_Package.Foreground_Color_Type;
                        Width            : in Natural) is

   Temp_Length: Natural :=
                MIN(Width,Variable_String_Package.Length_Of(The_Item.Item_VString));
   Temp_Text   : constant String := Variable_String_Package.Text_Of
                 (The_Item.Item_VString) (1 .. Temp_Length);

begin

   Screen_Package.Save_Cursor_Position;
     -- Set foreground and background colors for text
   Screen_Package.Set_Background_Color (Background_Color);
   Screen_Package.Set_Foreground_Color (Foreground_Color);
     -- print entire string
   Screen_Package.Put (Temp_Text);

     -- if there is a hot key position, overprint that character in the
     -- highlight color
   if (The_Item.Hot_Key_Position > 0) and (Screen_Package.Foreground_Color_Type'POS
      (Foreground_Color)/=Screen_Package.Foreground_Color_Type'POS (Highlight_Color))
   then
      Screen_Package.Set_Foreground_Color (Highlight_Color);
      Screen_Package.Restore_Cursor_Position;

      if (The_Item.Hot_Key_Position > 1) then
         Screen_Package.Move_Cursor_Forward (Screen_Package.Column_Type
         (The_Item.Hot_Key_Position - 1));
      end if;

      Screen_Package.Put (Temp_Text (The_Item.Hot_Key_Position));
   end if;
```

```
   Screen_Package.Restore_Cursor_Position;

end Display_Item;

-------------------------------------------------------------------------------
-------------------------------------------------------------------------------

  -- Quickly moves highlight bar to another position in current display window.
  -- NOTE: Assumes the old highlighted position and the new highlighted Current_Item
  -- are within the current display window, and does not perform any checking.
procedure Fast_Bar_Move (The_Menu : in out Menu_Type) is

   Temp_Window_Width : Natural := Menu_Window_Width (The_Menu);
   Temp_Item         : Menu_Item_Type;
   Temp_List_Index   : Natural;

begin


     -- Erase old highlight bar, if one exists
   if The_Menu.Highlight_Bar > 0 then

      Screen_Package.Position_Cursor (The_Menu.Upper_Left_Row +
                                      Screen_Package.Row_Type
                                     (The_Menu.Highlight_Bar -
                                      The_Menu.First_Window_Item + 1),
                                      The_Menu.Upper_Left_Column + 1);
      Temp_List_Index := The_Menu.Highlight_Bar;
      Menu_List_Package.Get_Item (The_Item => Temp_Item, From_List =>
                                  The_Menu.Item_List,
                                  At_Position => Temp_List_Index);
      Display_Item (The_Item         => Temp_Item,
                    Background_Color => The_Menu.Color_Scheme.Background_Color,
                    Foreground_Color => The_Menu.Color_Scheme.Foreground_Color,
                    Highlight_Color  => The_Menu.Color_Scheme.Highlight_Color,
                    Width            => Temp_Window_Width);
   end if;

     -- Write Current_Item in "reversed colors"
   Highlight_Row_Cursor_Position := The_Menu.Upper_Left_Row +
                                    Screen_Package.Row_Type (The_Menu.Current_Item -
                                    The_Menu.First_Window_Item + 1);
   Highlight_Column_Cursor_Position := The_Menu.Upper_Left_Column + 1;
   Screen_Package.Position_Cursor (Highlight_Row_Cursor_Position,
                                   Highlight_Column_Cursor_Position);
   Temp_List_Index := The_Menu.Current_Item;
   Menu_List_Package.Get_Item (The_Item => Temp_Item, From_List => The_Menu.Item_List,
                               At_Position => Temp_List_Index);
   Display_Item (The_Item         => Temp_Item,
                 Background_Color => Screen_Package.White_Background,
                 Foreground_Color => Screen_Package.Black_Foreground,
                 Highlight_Color  => Screen_Package.Black_Foreground,
                 Width            => Temp_Window_Width);
   The_Menu.Highlight_Bar := The_Menu.Current_Item;


end Fast_Bar_Move;

-------------------------------------------------------------------------------
-------------------------------------------------------------------------------

  -- Displays the menu items in the "current display window", i.e. those items
  -- in the item list postions First_Window_Item..Last_Window_Item.
```

```
  -- The Current_Item, if any, is displayed with the highlight bar on it.
procedure Display_Menu_Items (Of_Menu : in Menu_Type) is

   Temp_Window_Width: Natural := Menu_Window_Width (Of_Menu);
   Temp_Blank_String: constant String (1 .. Temp_Window_Width) := (others => ' ');
   Temp_Row         : Screen_Package.Row_Type            :=Of_Menu.Upper_Left_Row+1;
   Temp_Col         :constant Screen_Package.Column_Type:=Of_Menu.Upper_Left_Column+1;
   Temp_Item        : Menu_Item_Type;
   Temp_List_Index  : Natural;

begin

     -- "Blank" the window part of the menu
   Screen_Package.Set_Foreground_Color (Of_Menu.Color_Scheme.Foreground_Color);
   Screen_Package.Set_Background_Color (Of_Menu.Color_Scheme.Background_Color);

   for I in Of_Menu.Upper_Left_Row + 1 .. Of_Menu.Lower_Right_Row - 1 loop
      Screen_Package.Position_Cursor (I, Temp_Col);
      Screen_Package.Put (Temp_Blank_String);
   end loop;

   if Of_Menu.Item_List_Length = 0 then
      Highlight_Row_Cursor_Position    := Temp_Row;
      Highlight_Column_Cursor_Position := Temp_Col;
   else

       -- Display just the items currently in the window
      for I in Of_Menu.First_Window_Item .. Of_Menu.Last_Window_Item loop

         Temp_List_Index := I;
         Screen_Package.Position_Cursor (Temp_Row, Temp_Col);
         Menu_List_Package.Get_Item (The_Item => Temp_Item, From_List =>
                                     Of_Menu.Item_List, At_Position =>
                                     Temp_List_Index);

           -- Reverse video "BAR" will be positioned on "Current_Item"
         if I = Of_Menu.Current_Item then
          -- Save the highlighted row,column position for Read_Menu when it is called.
            Highlight_Row_Cursor_Position    := Temp_Row;
            Highlight_Column_Cursor_Position := Temp_Col;
            Display_Item (The_Item         => Temp_Item,
                          Background_Color => Screen_Package.White_Background,
                          Foreground_Color => Screen_Package.Black_Foreground,
                          Highlight_Color  => Screen_Package.Black_Foreground,
                          Width            => Temp_Window_Width);
            Of_Menu.Highlight_Bar := Of_Menu.Current_Item;

         else
            Display_Item (The_Item         => Temp_Item,
                          Background_Color => Of_Menu.Color_Scheme.Background_Color,
                          Foreground_Color => Of_Menu.Color_Scheme.Foreground_Color,
                          Highlight_Color  => Of_Menu.Color_Scheme.Highlight_Color,
                          Width            => Temp_Window_Width);
         end if;

         Temp_Row := Temp_Row + 1;

      end loop;

   end if;


end Display_Menu_Items;
```

```
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------

  -- Updates the display by moving the highlight bar with "Fast_Bar_Move" if the
  -- old and new bar positions are already on the screen. Redefines and displays
  -- the whole display window if some previously undisplayed items must be displayed.
procedure Update_Menu_Items (The_Menu : in out Menu_Type) is

begin

     -- Current item is in window already
   if Is_In_Window (The_Item_Number => The_Menu.Current_Item, Of_Menu => The_Menu)
   then
      Fast_Bar_Move (The_Menu => The_Menu);
   else    -- Current item not in the window
      Determine_Items_In_Window (The_Menu => The_Menu);
      Display_Menu_Items (Of_Menu => The_Menu);
   end if;

end Update_Menu_Items;

-- ==========================================================================

procedure Display_Menu (The_Menu : in Menu_Type) is

   Saved_Foreground_Color : Screen_Package.Foreground_Color_Type;
   Saved_Background_Color : Screen_Package.Background_Color_Type;

begin

   Screen_Package.Get_Screen;
   Saved_Foreground_Color := Screen_Package.Present_Foreground_Color;
   Saved_Background_Color := Screen_Package.Present_Background_Color;
   Internal_Draw_Border (The_Menu.Upper_Left_Row, The_Menu.Upper_Left_Column,
                         The_Menu.Lower_Right_Row, The_Menu.Lower_Right_Column,
                         The_Menu.Color_Scheme, Variable_String_Package.Text_Of
                        (The_Menu.Title), The_Menu.Border_Style);
   Display_Menu_Items (Of_Menu => The_Menu);
     -- Restore the original foreground and background colors.
   Screen_Package.Set_Foreground_Color (Saved_Foreground_Color);
   Screen_Package.Set_Background_Color (Saved_Background_Color);
   Screen_Package.Release_Screen;

end Display_Menu;

-------------------------------------------------------------------------------
-------------------------------------------------------------------------------

  -- Hides the menu by overwriting it with blanks.  Uses either the
  -- fill color passed in from the calling routine or the current
  -- background color.
procedure Hide_Menu (The_Menu   : in Menu_Type;
                     Fill_Color : in Screen_Package.Background_Color_Type) is

   Temp_Background_Color : Screen_Package.Background_Color_Type;
   Temp_String : constant String (integer (The_Menu.Upper_Left_Column) .. integer
                                  (The_Menu.Lower_Right_Column)) :=
                                  (others => ' ');

begin

   Screen_Package.Get_Screen;
```

```
   Temp_Background_Color := Screen_Package.Present_Background_Color;
   Screen_Package.Set_Background_Color (Fill_Color);

   for I in The_Menu.Upper_Left_Row .. The_Menu.Lower_Right_Row loop
      Screen_Package.Position_Cursor (I, The_Menu.Upper_Left_Column);
      Screen_Package.Put (Temp_String);
   end loop;

   Screen_Package.Set_Background_Color (Temp_Background_Color);

   Screen_Package.Release_Screen;

end Hide_Menu;

-- ==========================================================================

procedure Menu_Page_Down (The_Menu : in out Menu_Type) is
   Temp_Window_Length : Natural := Menu_Window_Length (The_Menu);
begin

     -- Move Current_Item down one page, or to bottom of list, whichever comes first.
   if (The_Menu.Current_Item + Temp_Window_length) >= The_Menu.Item_List_Length then
      The_Menu.Current_Item := The_Menu.Item_List_Length;
   else
      The_Menu.Current_Item := The_Menu.Current_Item + Temp_Window_Length;
   end if;

     -- Adjust the display window to include the new current item.
   Update_Menu_Items (The_Menu => The_Menu);

end Menu_Page_Down;

-------------------------------------------------------------------------------
-------------------------------------------------------------------------------

procedure Menu_Page_Up (The_Menu : in out Menu_Type) is
   Temp_Window_Length : Natural := Menu_Window_Length (The_Menu);
begin

     -- Move Current_Item up one page, or to top of list, whichever comes first.
   if The_Menu.Current_Item > Temp_Window_length then
      The_Menu.Current_Item := The_Menu.Current_Item - Temp_Window_Length;
   elsif The_Menu.Current_Item > 0 then
      The_Menu.Current_Item := 1;
   end if;

     -- Adjust the display window to include the new current item.
   Update_Menu_Items (The_Menu => The_Menu);

end Menu_Page_Up;

-- ==========================================================================

procedure Menu_First_Page (The_Menu : in out Menu_Type) is
begin

     -- Move Current_Item to top of list.
   The_Menu.Current_Item := 1;
     -- Adjust the display window to include the new current item.
   Update_Menu_Items (The_Menu => The_Menu);

end Menu_First_Page;
```

```
-- ==========================================================================

procedure Menu_Last_Page (The_Menu : in out Menu_Type) is
begin

     -- Move Current_Item to bottom of list.
   The_Menu.Current_Item := The_Menu.Item_List_Length;
     -- Adjust the display window to include the new current item.
   Update_Menu_Items (The_Menu => The_Menu);

end Menu_Last_Page;

-------------------------------------------------------------------------------
-------------------------------------------------------------------------------

procedure Menu_Bar_Down_One_Line (The_Menu : in out Menu_Type) is

   Temp_Window_Length    : Natural := Menu_Window_Length (The_Menu);
   Temp_Item_List_Length : Natural := Menu_List_Package.Length_Of
                                      (The_Menu.Item_List);

begin

     -- There are items, window space to display items, and there is a menu bar.
   if (Temp_Window_Length > 0) and (The_Menu.Current_Item > 0) and
      (Temp_Item_List_Length > 0) then

        -- Move Current_Item "cyclically" to next item of list.
      The_Menu.Current_Item := Next_Item (Of_Menu => The_Menu);
        -- Adjust the display window to include the new current item.
      Update_Menu_Items (The_Menu => The_Menu);

   end if;

end Menu_Bar_Down_One_Line;

-- ==========================================================================

procedure Menu_Bar_Up_One_Line (The_Menu : in out Menu_Type) is

   Temp_Item_List_Length : Natural := Menu_List_Package.Length_Of
                                     (The_Menu.Item_List);
   Temp_Window_Length    : Natural := Menu_Window_Length (The_Menu);

begin

     -- There are items, window space to display items, and there is a menu bar.
   if (Temp_Window_Length > 0) and (The_Menu.Current_Item > 0) and
      (Temp_Item_List_Length > 0) then

        -- Move Current_Item "cyclically" to previous item of list.
      The_Menu.Current_Item := Prev_Item (Of_Menu => The_Menu);
        -- Adjust the display window to include the new current item.
      Update_Menu_Items (The_Menu => The_Menu);

   end if;

end Menu_Bar_Up_One_Line;

-------------------------------------------------------------------------------
-------------------------------------------------------------------------------

  -- Returns the index of the next location of The_Char in In_String, starting
```

```
      -- from Starting_Pos.  Returns 0 if The_Char does not occur in the string
      -- or if the only occurance of The_Char is at the current position.
   function Find_Char_Index_In_String (The_Char     : in Character;
                                       In_String    : in String;
                                       Starting_Pos : in Natural) return Natural is
   begin

      for String_Index in Starting_Pos + 1 .. In_String'LAST loop
         if (The_Char = In_String (String_Index)) then
            return Natural (String_Index);
         end if;
      end loop;

      for String_Index in In_String'FIRST .. Starting_Pos - 1 loop
         if (The_Char = In_String (String_Index)) then
            return Natural (String_Index);
         end if;
      end loop;

      return 0;

   end Find_Char_Index_In_String;

   -- ==========================================================================

      -- Check to see if a special key is in the array of special keys
   function Check_Special_Keys (Legal_Special_Keys : in

Keyboard_Package.Special_Keys_Array_Type;
                                 Special_Key        : in
Keyboard_Package.Special_Keys_Type)
   return Boolean is
   begin -- Check_Value

      for Key in Legal_Special_Keys'RANGE loop
         if Convert_Special_Key_To_Char (Special_Key) = Convert_Special_Key_To_Char
           (Legal_Special_Keys (Key)) then
            return True;
         end if;
      end loop;

      return False;

   end Check_Special_Keys;

   -- ==========================================================================

   procedure Undefine_Menu (The_Menu : in out Menu_Type) Is
   begin

      if The_Menu /= null then
         Variable_String_Package.Destroy (The_Menu.Title);
         Dispose_Of_Menu( The_Menu);
      end if;

   end Undefine_Menu;

   ------------------------------------------------------------------------------
   procedure Undefine_Menu (The_Menu : in out Menu_Type;
                            The_List : in out Menu_List_Package.List_Type) is
   begin

      Menu_List_Package.Clear( The_List);
```

```
   Undefine_Menu (The_Menu);

end Undefine_Menu;

-------------------------------------------------------------------------------
-------------------------------------------------------------------------------

procedure Read_Menu (The_Menu   : in out Menu_Type;
                     The_Choice : out Natural) is

   Special_Key_List  : constant Keyboard_Package.Special_Keys_Array_Type :=
                               (Keyboard_Package.Home_Key,
                                Keyboard_Package.Up_Arrow_Key,
                                Keyboard_Package.Pg_Up_Key,
                                Keyboard_Package.Left_Arrow_Key,
                                Keyboard_Package.Right_Arrow_Key,
                                Keyboard_Package.End_Key,
                                Keyboard_Package.Down_Arrow_Key,
                                Keyboard_Package.Down_Arrow_Key,
                                Keyboard_Package.Pg_Dn_Key);

   Kbd_Char          : Character := ASCII.nul;
   Special_Key       : Keyboard_Package.Special_Keys_Type;
   Is_Special_Key    : Boolean          := false;
   Temp_Window_Width : constant Natural := Menu_Window_Width (The_Menu);
   Item_List_Length  : constant Natural := Menu_List_Package.Length_Of
                       (The_Menu.Item_List);
   Temp_List_Index   : Natural;
   Hot_Key_String    : String (1 .. Item_List_Length) := (others => ' ');
   Temp_Item         : Menu_Item_Type;
   Hot_Key_Item      : Natural := 0;

   Saved_Foreground_Color : Screen_Package.Foreground_Color_Type;
   Saved_Background_Color : Screen_Package.Background_Color_Type;

begin


   if The_Menu.Current_Item = 0 then
      The_Menu.Current_Item := 1;
   end if;

     -- Construct the "Hot-Key String" of characters that will cause cursor
     -- movement to specific items.
   for I in 1 .. Item_List_Length loop

      Temp_List_Index := I;
      Menu_List_Package.Get_Item (The_Item => Temp_Item, From_List =>
                                  The_Menu.Item_List,
                                  At_Position => Temp_List_Index);
        -- If item has a Hot_Key and it shows up in the window, then store it
        -- in the appropriate position in Hot_Key_String.
      if Natural( Temp_Item.Hot_Key_Position) in 1 .. Temp_Window_Width then
         Hot_Key_String (I) := Variable_String_Package.Text_Of
                               (Temp_Item.Item_VString)
         (Positive (Temp_Item.Hot_Key_Position));
      end if;

   end loop;


   loop
        -- Valid inputs are:
```

```
  --   SPECIAL KEYS: Menu   => (Home_Key,Up_Arrow_Key,Pg_Up_Key,Left_Arrow_Key,
  --                  Right_Arrow_Key,End_Key,Down_Arrow_Key,Pg_Dn_Key)
  --   HOT KEYS:      Menu item characters denoted by user for fast cursor
  --                  movement.
  --   ENTER KEY:     Key that causes a selection to be made.


Keyboard_Package.Get_One_Valid_Char (Legal_Values        =>
                                                   Hot_Key_String&ASCII.CR,
                                     Legal_Special_Keys  => Special_Key_List,
                                     Special_Key_Hit     => Is_Special_Key,
                                     Entered_Special_Key => Special_Key,
                                     Entered_Value       => Kbd_Char);

if The_Menu.Item_List_Length = 0 then
   exit when Kbd_Char = ascii.cr;
elsif Is_Special_Key then

   -- Actions to take for the special keys.
   Screen_Package.Get_Screen;
   Saved_Foreground_Color := Screen_Package.Present_Foreground_Color;
   Saved_Background_Color := Screen_Package.Present_Background_Color;
   Screen_Package.Position_Cursor (Highlight_Row_Cursor_Position,
                                   Highlight_Column_Cursor_Position);

   case Special_Key is
      when Keyboard_Package.Up_Arrow_Key | Keyboard_Package.Left_Arrow_Key =>
         Menu_Bar_Up_One_Line (The_Menu => The_Menu);
      when Keyboard_Package.Down_Arrow_Key | Keyboard_Package.Right_Arrow_Key =>
         Menu_Bar_Down_One_Line (The_Menu => The_Menu);
      when Keyboard_Package.Home_Key =>
         Menu_First_Page (The_Menu => The_Menu);
      when Keyboard_Package.End_Key =>
         Menu_Last_Page (The_Menu => The_Menu);
      when Keyboard_Package.Pg_Up_Key =>
         Menu_Page_Up (The_Menu => The_Menu);
      when Keyboard_Package.Pg_Dn_Key =>
         Menu_Page_Down (The_Menu => The_Menu);
      when others =>
         null;
   end case;

   -- Restore the original foreground and background colors.
   Screen_Package.Set_Foreground_Color (Saved_Foreground_Color);
   Screen_Package.Set_Background_Color (Saved_Background_Color);
   Screen_Package.Release_Screen;


else

     -- a carriage return has been entered
     -- The highlighted Current_Item has been selected.
   exit when Kbd_Char = ascii.cr;
     -- Hot_Key was pressed. Move highlight bar to the corresponding item.
     -- Key pressed is a character. is it a Hot_Key or a carriage return?
   Hot_Key_Item := Find_Char_Index_In_String( The_Char    => Kbd_Char,
                                              In_String   => Hot_Key_String,
                                              Starting_Pos=>
                                                     The_Menu.Current_Item);
   if Hot_Key_Item > 0 then

      -- Actions to take for the special keys.
      Screen_Package.Get_Screen;
```

```
                Saved_Foreground_Color := Screen_Package.Present_Foreground_Color;
                Saved_Background_Color := Screen_Package.Present_Background_Color;

                The_Menu.Current_Item := Hot_Key_Item;
                Update_Menu_Items (The_Menu => The_Menu);

                -- Restore the original foreground and background colors.
                Screen_Package.Set_Foreground_Color (Saved_Foreground_Color);
                Screen_Package.Set_Background_Color (Saved_Background_Color);
                Screen_Package.Release_Screen;

             end if;

          end if;

       end loop;

       The_Choice := The_Menu.Current_Item;


    end Read_Menu;

    ------------------------------------------------------------------------------
    ------------------------------------------------------------------------------

      -- Allows the user to make a selection from the menu.
    procedure Read_Menu (The_Menu            : in out Menu_Type;
                         Legal_Special_Keys  : in
 Keyboard_Package.Special_Keys_Array_Type;
                         Special_Key_Hit     : out Boolean;
                         Entered_Special_Key : out Keyboard_Package.Special_Keys_Type;
                         Escape_Key_Hit      : out Boolean;
                         The_Choice          : out Natural) is

       use Keyboard_Package;

       Menu_Keys            : constant Keyboard_Package.Special_Keys_Index_Type := 9;
       Special_Key_List     : Keyboard_Package.Special_Keys_Array_Type
                             (Keyboard_Package.Special_Keys_Index_Type'first ..
                                Menu_Keys + Keyboard_Package.Special_Keys_Index_Type'pos (
                                Legal_Special_Keys'last));
       Menu_Key_List        : Keyboard_Package.Special_Keys_Array_Type
                              (Keyboard_Package.Special_Keys_Index_Type'first ..
                               Menu_Keys) :=
                              (Keyboard_Package.Home_Key, Keyboard_Package.Up_Arrow_Key,
                               Keyboard_Package.Pg_Up_Key,Keyboard_Package.Left_Arrow_Key,
                               Keyboard_Package.Right_Arrow_Key, Keyboard_Package.End_Key,
                               Keyboard_Package.Down_Arrow_Key,
                               Keyboard_Package.Down_Arrow_Key,
                               Keyboard_Package.Pg_Dn_Key);
       Kbd_Char              : Character;
       Special_Key           : Keyboard_Package.Special_Keys_Type;
       Is_Special_Key        : Boolean          := false;
       Temp_Window_Width     : constant Natural := Menu_Window_Width (The_Menu);
       Item_List_Length      : constant Natural := Menu_List_Package.Length_Of
                                                     (The_Menu.Item_List);
       Temp_List_Index       : Natural;
       Hot_Key_String        : String (1 .. Item_List_Length) := (others => ' ');
       Temp_Item             : Menu_Item_Type;
       Hot_Key_Item          : Natural := 0;

       Saved_Foreground_Color : Screen_Package.Foreground_Color_Type;
       Saved_Background_Color : Screen_Package.Background_Color_Type;
```

```
begin

   Special_Key_Hit := FALSE;
   Escape_Key_Hit  := FALSE;

   if The_Menu.Current_Item = 0 then
      The_Menu.Current_Item := 1;
   end if;

     -- Construct the "Hot-Key String" of characters that will cause cursor
     -- movement to specific items.
   for I in 1 .. Item_List_Length loop

      Temp_List_Index := I;
      Menu_List_Package.Get_Item (The_Item    => Temp_Item,
                                  From_List   => The_Menu.Item_List,
                                  At_Position => Temp_List_Index);
        -- If item has a Hot_Key and it shows up in the window, then store it
        -- in the appropriate position in Hot_Key_String.
      if Natural( Temp_Item.Hot_Key_Position) in 1 .. Temp_Window_Width then
         Hot_Key_String (I) := Variable_String_Package.Text_Of
                               (Temp_Item.Item_VString)
                               (Positive (Temp_Item.Hot_Key_Position));
      end if;

   end loop;

     -- Construct the special key list.
   Special_Key_List (Keyboard_Package.Special_Keys_Index_Type'first .. Menu_Keys) :=
                     Menu_Key_List;
   Special_Key_List (Menu_Keys + 1 .. Menu_Keys +
                     Keyboard_Package.Special_Keys_Index_Type'pos
                    (Legal_Special_Keys'last)) :=
            Legal_Special_Keys;

   loop
      -- Valid inputs are:
      --   SPECIAL KEYS: Menu   => (Home_Key,Up_Arrow_Key,Pg_Up_Key,Left_Arrow_Key,
      --                  Right_Arrow_Key,End_Key,Down_Arrow_Key,Pg_Dn_Key)
      --                  Also, any special keys specified in the Legal_Special_Keys -
      --                 array
      --                  by the user (if the predefined ones appear in this array, it
      --                  will be ignored).
      --   HOT KEYS:      Menu item characters denoted by user for fast cursor
      --                  movement.
      --   ENTER KEY:     Key that causes a selection to be made.
      --   ESCAPE KEY:    Often used as a special-purpose key during user input (sets
      --                  Escape_Key_Hit flag parameter).
    Keyboard_Package.Get_One_Valid_Char
                                  (Legal_Values=>Hot_Key_String&ascii.CR&ASCII.ESC,
                                           Legal_Special_Keys  => Special_Key_List,
                                           Special_Key_Hit     => Is_Special_Key,
                                           Entered_Special_Key => Special_Key,
                                           Entered_Value       => Kbd_Char);

      if The_Menu.Item_List_Length = 0 then
         Escape_Key_Hit := Kbd_Char = ASCII.Esc;  -- user pressed the escape key
         exit when Kbd_Char = Ascii.CR or Kbd_Char = ASCII.Esc;

      elsif Is_Special_Key then

           -- Actions to take for the special keys.
```

```
   Screen_Package.Get_Screen;
   Saved_Foreground_Color := Screen_Package.Present_Foreground_Color;
   Saved_Background_Color := Screen_Package.Present_Background_Color;
   Screen_Package.Position_Cursor (Highlight_Row_Cursor_Position,
                                   Highlight_Column_Cursor_Position);

   case Special_Key is
      when Keyboard_Package.Up_Arrow_Key | Keyboard_Package.Left_Arrow_Key =>
         Menu_Bar_Up_One_Line (The_Menu => The_Menu);
      when Keyboard_Package.Down_Arrow_Key | Keyboard_Package.Right_Arrow_Key =>
         Menu_Bar_Down_One_Line (The_Menu => The_Menu);
      when Keyboard_Package.Home_Key =>
         Menu_First_Page (The_Menu => The_Menu);
      when Keyboard_Package.End_Key =>
         Menu_Last_Page (The_Menu => The_Menu);
      when Keyboard_Package.Pg_Up_Key =>
         Menu_Page_Up (The_Menu => The_Menu);
      when Keyboard_Package.Pg_Dn_Key =>
         Menu_Page_Down (The_Menu => The_Menu);
      when others =>
         if Check_Special_Keys (Legal_Special_Keys, Special_Key) then
            Special_Key_Hit       := TRUE;
            Entered_Special_Key   := Special_Key;
            Screen_Package.Release_Screen;
            exit;
         end if;
   end case;

   -- Restore the original foreground and background colors.
   Screen_Package.Set_Foreground_Color (Saved_Foreground_Color);
   Screen_Package.Set_Background_Color (Saved_Background_Color);
   Screen_Package.Release_Screen;



else

   if Kbd_Char = ASCII.CR then
      exit;      -- The highlighted Current_Item has been selected.

   elsif Kbd_Char = ASCII.ESC then
      Escape_Key_Hit := True;      -- The user pressed the escape key
      exit;

   else      -- Key pressed is a character. is it a Hot_Key?
      Hot_Key_Item := Find_Char_Index_In_String (The_Char => Kbd_Char,
                                                 In_String=> Hot_Key_String,
                                                 Starting_Pos =>
                                                      The_Menu.Current_Item);
      if (Hot_Key_Item > 0) then
         -- Hot_Key was pressed. Move highlight bar to the corresponding item.

         -- Actions to take for the special keys.
         Screen_Package.Get_Screen;
         Saved_Foreground_Color := Screen_Package.Present_Foreground_Color;
         Saved_Background_Color := Screen_Package.Present_Background_Color;
         Screen_Package.Position_Cursor (Highlight_Row_Cursor_Position,
                                         Highlight_Column_Cursor_Position);

         The_Menu.Current_Item := Hot_Key_Item;
         Update_Menu_Items (The_Menu => The_Menu);

         -- Restore the original foreground and background colors.
```

```
                  Screen_Package.Set_Foreground_Color (Saved_Foreground_Color);
                  Screen_Package.Set_Background_Color (Saved_Background_Color);
                  Screen_Package.Release_Screen;

               end if;
            end if;

         end if;

      end loop;

      The_Choice := The_Menu.Current_Item;


   end Read_Menu;

end Menu_Package;
```
•

```
-------------------------------------------------------------------------------
-- File Name:       DOL_CMDS.MNU
-- Organization:    NASA - Langley Research Center (LaRC)
-- Project:         LIDAR In-Space Technology Experiment (LITE)
-------------------------------------------------------------------------------
--*****************************************************************************
-- Name/Number:
--    DOL_CMDS.MNU                      (Master AT Menu File)
--
-- Abstract:
--    This is the menu of all legal DOL commands that can be sent over DOLs
--    7-12.  Note that the entries in this menu **MUST** be in the same order
--    as the entries in the data file DOL_CMDS.DAT for the DOL interface to
--    work properly.  The strings themselves don't have to match, or even be
--    close, but each DOL command name in the data file should be in tha
--    corresponding location with the identifier for it in this menu.  See
--    comments at the top of the data file for more info.
--
-- Called By:
--    DOL_CMDS.ADB
--
--*****************************************************************************

[]                        -- Menu Title
[1]                       -- initial value of Current_Item
[15]                      -- Upper Left Row
[2]                       -- Upper Left Column
[35]                      -- Lower Right Row
[25]                      -- Lower Right Column
[Blue_Background]         -- Background Color
[Bright_White_Foreground] -- Foreground Color
[Yellow_Foreground]       -- Title Color
[Yellow_Foreground]       -- Border Color
[Light_Red_Foreground]    -- Highlight (Hot Key) Color
[Single]                  -- Border Type


************************* Menu Items ****************************

[7]
[Turn "Q" Switch On    ]

[7]
[Turn "Q" Switch Off   ]

[1]
[Flashlamps On         ]

[1]
[Flashlamps Off        ]

[8]
[Select Laser A        ]

[8]
[Select Laser B        ]

[1]
[Initialize Encoders   ]

[6]
[Stow Prism            ]
```

```
[1]
[Search                ]

[1]
[Align                 ]

[1]
[Last Aligned Position ]

[1]
[Camera On             ]

[1]
[Camera Off            ]

[7]
[Go To Day Data Take   ]

[7]
[Go To Night Data Take ]

[7]
[Go To Standby         ]

[7]
[Go To BITS            ]

[1]
[Ignore SDIO Commands  ]

[1]
[Accept SDIO Commands  ]
•
```

Appendix 2: Filename Manager Package

```
-------------------------------------------------------------------------------
-- File Name:       FNAMEMGR.ADS
-- Organization:    NASA - Langley Research Center (LaRC)
-- Project:         LIDAR In-Space Technology Experiment (LITE)
-------------------------------------------------------------------------------
--*****************************************************************************
-- Name/Number:
--    Filename_Manager_Package                                  (package spec)
--
-- Abstract:
--    This package is used to allow greater flexibility in naming external
--    files such as data files, initialization files, menu definition files,
--    and so on.  It allows the code to use identifier strings, called "tags",
--    to select files, instead of being limited to the file naming restrictions
--    imposed by the host operating system.  In addition, it also prevents
--    literal file names from being "hard-wired" into the code.  This allows
--    the programmer to change file names, or use a substitute file with a
--    different name in a test, without having to modify or re-compile his
--    code.
--
-- Acronyms/Abbreviations:
--    None
--
-- Dependencies:
--    None
--
-- Global Objects:
--    None
--
-- Exceptions:
--    Duplicate_Filename
--    Duplicate_Tag
--    Invalid_File
--    Invalid_Filename
--    Invalid_File_Contents
--    Invalid_Position
--    Invalid_Tag
--
-- Machine/Compiler Dependencies:
--    None
--
--*****************************************************************************

WITH Variable_String_Package;
PACKAGE Filename_Manager_Package IS

   -------------------------------------------------------------------------------
   --
   -- VISIBLE TYPES
   --
   -------------------------------------------------------------------------------
   -- file name/tag list type
   TYPE Filename_List_Type IS PRIVATE;

   -- When writing a filename list to an external file, this enumerated type
   -- allows a calling routine to have control over the desired type of
   -- file I/O action.  When generating a filename list from an external file
   -- it allows the calling routine have control over the type of action
   -- taken on the input list.
   TYPE List_Action_Type IS (Overwrite,
                             Append);
```

```
-- This enumerated type is used in the procedure that Sorts a list.
-- It allows the calling routine the choice of reordering the
-- list either by tag or name.
TYPE List_Sort_Type IS (By_File_Tag,
                        By_Filename);



--------------------------------------------------------------------------
--
-- VISIBLE EXCEPTIONS
--
--------------------------------------------------------------------------
-- When searching a list for a file tag, this exception is raised if the
-- file tag is not found
Invalid_Tag : EXCEPTION;

-- When searching a list for a filename, this exception is raised if the
-- filename is not found
Invalid_Filename : EXCEPTION;

-- This exception is raised when a new file tag to be added to a
-- filename list already exists in that list.
Duplicate_Tag : EXCEPTION;

-- This exception is raised when a new filename to be added to a
-- filename list already exists in that list.
Duplicate_Filename : EXCEPTION;

-- This exception is raised when a list position passed to a function does
-- not exist.
Invalid_Position : EXCEPTION;

-- This exception is raised when a problem is encountered with opening
-- or creating an external file
Invalid_File : EXCEPTION;

-- This exception is raised when a problem is encounte red while getting
-- token pairs from an external file.  The problem may be a token that
-- is too long ( > 80 chars) or a token may be missing
Invalid_File_Contents : EXCEPTION;



--------------------------------------------------------------------------
--
-- VISIBLE FUNCTIONS
--
--------------------------------------------------------------------------
--------------------------------------------------------------------------
--
-- This function returns the number of nodes in a filename list.
--
-- Inputs:
--    Filename_List : input list whose number of nodes (tag/filename pairs)
--                    is to be returned
-- Outputs:
--
-- Exceptions:
--
--------------------------------------------------------------------------
FUNCTION Length_Of
```

```
            (Filename_List : IN     Filename_List_Type)  RETURN NATURAL;


--------------------------------------------------------------------------
--------------------------------------------------------------------------
--
-- This procedure clears (deallocates) the input list
--
-- Inputs:
--    Filename_List : input list to be cleared
--
-- Outputs:
--    Filename_List : cleared null list to return
--
-- Exceptions:
--
--------------------------------------------------------------------------
PROCEDURE Clear_Filename_List (Filename_List : IN OUT Filename_List_Type);



--------------------------------------------------------------------------
--------------------------------------------------------------------------
--
-- This procedure allows the user to save the current contents of a
-- filename list to an external file.  If the Action is "Append",
-- then the tag/filename pairs in the list are appended to the end of the
-- To_File, empty or not.  In this case, if the input list happens to
-- be null, the file will remain unchanged.  If the Action is "Overwrite"
-- and the file is not empty, then the tag/filename pairs will overwrite
-- the contents of the file.  In this case, if the input list happens
-- to be null, the file will be empty afterwords.
--
-- Inputs:
--    Filename_List - tag/filename list to be written from
--    To_File       - external output file for which to write the
--                    tag/filename pairs
--    Action        - action to take on file, either append or overwrite
--
-- Outputs:
--
-- Exceptions:
--    Invalid_File  - raised when there is a problem with the external
--                    file open or creation
--------------------------------------------------------------------------
PROCEDURE Save_Filename_List (Filename_List : IN     Filename_List_Type;
                              To_File       : IN     STRING;
                              Action        : IN     List_Action_Type);



--------------------------------------------------------------------------
--------------------------------------------------------------------------
--
-- This procedure is used to create a new list of filenames and tags.
-- It reads the filenames and the tags
-- from an external file. If the Action is
-- "Overwrite", the list will be overwritten with the new entries found in
-- the From_File.  If the Action is Append, the tag/filename pairs found
-- in the given From_File will be appended to the list.  The From_File
-- is checked for duplicate file tags and filenames and exceptions raised
-- when necessary.
```

```
--
-- Inputs:
--    Filename_List - tag/filename list to be manipulated
--    From_File     - external file from which to read the tag/filename
--                    pairs and put int the Filename_List
--    Action        - action to take on list, either append or overwrite
--
-- Outputs:
--    Filename_List - tag/filename list to be manipulated
--
-- Exceptions:
--    Invalid_File_Contents - raised when an expected token can not
--                            be found in the external file or
--                            if a token is too long ( > Max_Token_Length)
--    Invalid_File          - raised when there is a problem with the
--                            external file open or creation
-------------------------------------------------------------------------------
PROCEDURE Generate_Filename_List
             (Filename_List : IN OUT Filename_List_Type;
              From_File     : IN     STRING;
              Action        : IN     List_Action_Type);


-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
--
-- This procedure allows the user to add a new file tag/filename pair to
-- the end of the filename list.  Duplicate tags and filenames are not
-- allowed and raise the Duplicate_Tag or Duplicate_Filename exceptions.
--
-- Inputs:
--    Filename_List - tag/filename list to be add to
--    File_Tag      - file tag to be added to list end
--    Filename      - filename to be added to list end
--
-- Outputs:
--    Filename_List - tag/filename list after addition of new pair
--
-- Exceptions:
--    Duplicate_Filename - raised when the input filename is found to
--                         already exist in the list
--    Duplicate_File_Tag - raised when the input file tag is found to
--                         already exist in the list
-------------------------------------------------------------------------------
PROCEDURE Append_To_Filename_List
             (Filename_List : IN OUT Filename_List_Type;
              File_Tag      : IN     STRING;
              Filename      : IN     STRING);



-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
--
-- This procedure allows the user to do a "sorted" insert of a new
-- file tag/filename pair into the filename list.  The pair will be
-- alphanumerically inserted either by file tag or by filename.  The
-- routine assumes that the list is already properly sorted for the
-- desired insert.  Results will be unpredictable if either the list
-- is not already sorted or if the pair is inserted by the wrong
-- token (tag or name).  Duplicate tags and filenames are not
-- allowed and raise the Duplicate_Tag or Duplicate_Filename exceptions.
--
```

```
-- Inputs:
--    Filename_List - tag/filename list to be add to
--    File_Tag      - file tag to be added to list end
--    Filename      - filename to be added to list end
--    Insert        - insert type (by file tag or filename)
--
-- Outputs:
--    Filename_List - tag/filename list after addition of new pair
--
-- Exceptions:
--    Duplicate_Filename - raised when the input filename is found to
--                         already exist in the list
--    Duplicate_File_Tag - raised when the input file tag is found to
--                         already exist in the list
-----------------------------------------------------------------------------
PROCEDURE Insert_Into_Filename_List
             (Filename_List : IN OUT Filename_List_Type;
              File_Tag      : IN     STRING;
              Filename      : IN     STRING;
              Insert        : IN     List_Sort_Type);


-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
--
-- This procedure allows the user to delete a file tag/filename pair
-- from a filename list.  The list is searched for the given file tag and
-- when found, the tag/filename pair are deleted.  The Invalid_Tag
-- exception will be raised if the file tag is not found.
--
-- Inputs:
--    Filename_List - tag/filename list to be deleted from
--    File_Tag      - file tag to delete (filename deleted also)
--
-- Outputs:
--    Filename_List - tag/filename list after addition of new pair
--
-- Exceptions:
--    Invalid_Tag - raised when the input file tag is not found in the list
-----------------------------------------------------------------------------
PROCEDURE Delete_File_Tag
             (Filename_List : IN OUT Filename_List_Type;
              File_Tag      : IN     STRING);



-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
--
-- This procedure allows the user to delete a file tag/filename pair
-- from a filename list.  The list is searched for the given filename and
-- when found, the tag/filename pair are deleted.  The Invalid_Filename
-- exception will be raised if the filename is not found.
--
-- Inputs:
--    Filename_List - tag/filename list to be deleted from
--    Filename      - filename to delete (file tag deleted also)
--
-- Outputs:
--    Filename_List - tag/filename list after addition of new pair
--
-- Exceptions:
--    Invalid_Filename - raised when the input filename is not found
```

```
--                        in the list
-----------------------------------------------------------------------------
PROCEDURE Delete_Filename
             (Filename_List : IN OUT Filename_List_Type;
              Filename      : IN     STRING);



-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
--
-- This function searches a filename list for a filename and returns
-- a boolean telling whether or not it was found.
--
-- Inputs:
--    Filename_List - tag/filename list to be searched
--    Filename      - filename to search for in list
--
-- Outputs:
--
-- Exceptions:
-----------------------------------------------------------------------------
FUNCTION Filename_Exists
            (Filename_List : IN     Filename_List_Type;
             Filename      : IN     STRING)  RETURN BOOLEAN;



-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
--
-- This function searches a filename list for a file tag and returns
-- a boolean telling whether or not it was found.
--
-- Inputs:
--    Filename_List - tag/filename list to be searched
--    File_Tag      - tag to search for in list
--
-- Outputs:
--
-- Exceptions:
-----------------------------------------------------------------------------
FUNCTION File_Tag_Exists
            (Filename_List : IN     Filename_List_Type;
             File_Tag      : IN     STRING)  RETURN BOOLEAN;



-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
--
-- This function searches the specified list for the file tag passed
-- to it and returns the file name associated with that tag.  The
-- tag comparisons are NOT case sensitive.
--
-- Inputs:
--    Filename_List - tag/filename list to be searched
--    File_Tag      - tag to search for in list
--
-- Outputs:
--
-- Exceptions:
--    Invalid_Tag   - raised when the input file tag is not found
```

```
--                     in the list
-----------------------------------------------------------------------------
FUNCTION Get_Filename
            (Filename_List : IN     Filename_List_Type;
             File_Tag      : IN     STRING)  RETURN STRING;


-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
--
-- This function searches the specified list for the filename passed
-- to it and returns the file tag associated with that filename.  The
-- filename comparisons are NOT case sensitive.
--
-- Inputs:
--    Filename_List - tag/filename list to be searched
--    Filename      - filename to search for in list
--
-- Outputs:
--
-- Exceptions:
--    Invalid_Filename - raised when the input filename is not found
--                       in the list
-----------------------------------------------------------------------------
FUNCTION Get_File_Tag
            (Filename_List : IN     Filename_List_Type;
             Filename      : IN     STRING)  RETURN STRING;


-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
--
-- This function finds the node specified by the position passed
-- in and returns the filename at that position.
--
-- Inputs:
--    Filename_List - tag/filename list to be searched
--    Position      - position in list to return filename from
--
-- Outputs:
--
-- Exceptions:
--    Invalid_Position - position does not exist for input list
-----------------------------------------------------------------------------
FUNCTION Get_Filename
            (Filename_List : IN     Filename_List_Type;
             Position      : IN     Natural)  RETURN STRING;


-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
--
-- This function finds the node specified by the position passed
-- in and returns the file tag at that position.
--
-- Inputs:
--    Filename_List - tag/filename list to be searched
--    Position      - position in list to return file tag from
--
-- Outputs:
--
-- Exceptions:
--    Invalid_Position - position does not exist for input list
```

```
-----------------------------------------------------------------------------
FUNCTION Get_File_Tag
            (Filename_List : IN     Filename_List_Type;
             Position      : IN     Natural)  RETURN STRING;


-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
--
-- This function searches for the specified filename in the list, and
-- returns the position in the list of the node which contains it.
--
-- Inputs:
--    Filename_List - tag/filename list to be searched
--    Filename      - filename to search for in list
--
-- Outputs:
--
-- Exceptions:
--    Invalid_Filename - raised when the input filename is not in the
--                       input list
-----------------------------------------------------------------------------
FUNCTION Get_Filename_Position
            (Filename_List : IN     Filename_List_Type;
             Filename      : IN     STRING)  RETURN NATURAL;


-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
--
-- This function searches for the specified file tag in the list, and
-- returns the position in the list of the node which contains it.
--
-- Inputs:
--    Filename_List - tag/filename list to be searched
--    File_Tag      - file tag to search for in list
--
-- Outputs:
--
-- Exceptions:
--    Invalid_File_Tag - raised when the input file tag is not in the
--                       input list
-----------------------------------------------------------------------------
FUNCTION Get_File_Tag_Position
            (Filename_List : IN     Filename_List_Type;
             File_Tag      : IN     STRING) RETURN NATURAL;


-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
--
-- This procedure sorts a list.  The user has the option to do the
-- reorder by file tag or filename.  The procedure uses a bubble sort
-- swapping pointers when necessary instead of exchanging file tags and
-- filenames.
--
-- Inputs:
--    Filename_List - tag/filename list to be sorted
--    Sort          - sort type (by file tag or filename)
--
-- Outputs:
--    Filename_List - tag/filename list to be sorted
--
```

```
   -- Exceptions:
   ---------------------------------------------------------------------------
   PROCEDURE Sort_Filename_List
                (Filename_List : IN OUT Filename_List_Type;
                 Sort          : IN     List_Sort_Type);


   ---------------------------------------------------------------------------
   ---------------------------------------------------------------------------

   PRIVATE

      TYPE File_Node_Type;
      TYPE File_Node_Access_Type IS ACCESS File_Node_Type;
      TYPE File_Node_Type IS
         RECORD
            File_Tag        : Variable_String_Package.VString;
            Filename        : Variable_String_Package.VString;
            Next            : File_Node_Access_Type;
         END RECORD;

      TYPE Filename_List_Type IS
         RECORD
            Length    : NATURAL := 0;
            Node_List : File_Node_Access_Type;
         END RECORD;

END Filename_Manager_Package;
```

```
-----------------------------------------------------------------------------
-- File Name:       FNAMEMGR.ADB
-- Organization:    NASA - Langley Research Center (LaRC)
-- Project:         LIDAR In-Space Technology Experiment (LITE)
-----------------------------------------------------------------------------
--****************************************************************************
-- Name/Number:
--    Filename_Manager_Package                                  (package body)
--
-- Abstract:
--    This package is used to allow greater flexibility in naming external
--    files such as data files, initialization files, menu definition files,
--    and so on.  It allows the code to use identifier strings, called "tags",
--    to select files, instead of being limited to the file naming restrictions
--    imposed by the host operating system.  In addition, it also prevents
--    literal file names from being "hard-wired" into the code.  This allows
--    the programmer to change file names, or use a substitute file with a
--    different name in a test, without having to modify or re-compile his
--    code.
--
-- Acronyms/Abbreviations:
--    None
--
-- Dependencies:
--
--
-- Global Objects:
--    None
--
-- Exceptions:
--    Duplicate_Filename
--    Duplicate_Tag
--    Invalid_File
--    Invalid_Filename
--    Invalid_File_Contents
--    Invalid_Position
--    Invalid_Tag
--
-- Machine/Compiler Dependencies:
--    None
--
--****************************************************************************



WITH Unchecked_Deallocation;
WITH File_Utilities_Package;
PACKAGE BODY Filename_Manager_Package IS

   Max_Token_Length    : CONSTANT := 80;

   ---------------------------------------------------------------------------
   ---------------------------------------------------------------------------
   --
   -- Unchecked deallocation procedure
   ---------------------------------------------------------------------------
   PROCEDURE Dispose IS NEW Unchecked_Deallocation
                               (File_Node_Type, File_Node_Access_Type);



   ---------------------------------------------------------------------------
   ---------------------------------------------------------------------------
```

```
--
-- This internal function takes an input string, converts it to all
-- upper case, and then returns the new string.
--
-- Inputs:
--    Input_String - string to convert to upper case
--
-- Outputs:
--
-- Exceptions:
--
------------------------------------------------------------------------------
FUNCTION Convert_To_Upper(Input_String : IN     STRING) RETURN STRING IS

   Temp_String : STRING(1..Max_Token_Length) := (OTHERS => ' ');

   Lower_To_Upper_ASCII_Offset : CONSTANT
                               := CHARACTER'POS('a') - CHARACTER'POS('A');

BEGIN
   Temp_String(Input_String'RANGE) := Input_String;

   -- examine every character in the string
   FOR I IN Input_String'RANGE LOOP

      -- if lower case, convert it to upper case
      IF Temp_String(I) IN 'a'..'z' THEN
         Temp_String(I) := CHARACTER'VAL((CHARACTER'POS(Temp_String(I)) -
                              Lower_To_Upper_ASCII_Offset));
      END IF;
   END LOOP;

   RETURN Temp_String(Input_String'RANGE);

-- allow calling routine to handle the exceptions
EXCEPTION
   WHEN OTHERS =>
      RAISE;

END Convert_To_Upper;


------------------------------------------------------------------------------
------------------------------------------------------------------------------
--
-- This function returns the number of nodes in a filename list.
--
-- Inputs:
--    Filename_List : input list whose number of nodes (tag/filename pairs)
--                    is to be returned
-- Outputs:
--
-- Exceptions:
--
------------------------------------------------------------------------------
FUNCTION Length_Of
            (Filename_List : IN     Filename_List_Type) RETURN NATURAL IS
BEGIN
   RETURN Filename_List.Length;

-- allow calling routine to handle the exceptions
EXCEPTION
   WHEN OTHERS =>
```

```
      RAISE;

END Length_Of;


-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
--
-- This procedure clears (deallocates) the input list
--
-- Inputs:
--    Filename_List : input list to be cleared
--
-- Outputs:
--    Filename_List : cleared null list to return
--
-- Exceptions:
--
-----------------------------------------------------------------------------
PROCEDURE Clear_Filename_List(Filename_List : IN OUT Filename_List_Type) IS

   -- temporary pointer used to manipulate tag/filename list
   Temp_Ptr : File_Node_Access_Type := Filename_List.Node_List;

BEGIN
   -- delete every node
   WHILE Filename_List.Node_List /= NULL LOOP

      -- dispose of the tag and filename
      Variable_String_Package.
         Destroy (The_String => Temp_Ptr.File_Tag);
      Variable_String_Package.
         Destroy (The_String => Temp_Ptr.Filename);

      -- set the temp pointer to the next node
      Temp_Ptr := Temp_Ptr.Next;

      -- dispose of the current node
      Dispose (Filename_List.Node_List);

      -- make list point to next node
      Filename_List.Node_List := Temp_Ptr;
   END LOOP;

   -- reset the list length
   Filename_List.Length := 0;

-- allow calling routine to handle the exceptions
EXCEPTION
   WHEN OTHERS =>
      RAISE;

END Clear_Filename_List;



-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
--
-- This procedure allows the user to save the current contents of a
-- filename list to an external file.  If the Action is "Append",
-- then the tag/filename pairs in the list are appended to the end of the
-- To_File, empty or not.  In this case, if the input list happens to
```

```
-- be null, the file will remain unchanged.  If the Action is "Overwrite"
-- and the file is not empty, then the tag/filename pairs will overwrite
-- the contents of the file.  In this case, if the input list happens
-- to be null, the file will be empty afterwords.
--
-- Inputs:
--    Filename_List - tag/filename list to be written from
--    To_File       - external output file for which to write the
--                    tag/filename pairs
--    Action        - action to take on file, either append or overwrite
--
-- Outputs:
--
-- Exceptions:
--    Invalid_File  - raised when there is a problem with the external
--                    file open or creation
-------------------------------------------------------------------------------
PROCEDURE Save_Filename_List (Filename_List : IN     Filename_List_Type;
                              To_File       : IN     STRING;
                              Action        : IN     List_Action_Type) IS

   -- for opening file
   Aborted    : BOOLEAN               := FALSE;

   -- file utilities package file identifier
   File_ID    : File_Utilities_Package.File_Identifier_Type;

   -- temporary pointer used to manipulate tag/filename list
   Temp_Ptr   : File_Node_Access_Type := Filename_List.Node_List;

BEGIN
   -- open the file according to the desired action
   IF Action = Append THEN
      File_Utilities_Package.
         Open_Output_File
            (Filename   => To_File,
             File_Mode  => File_Utilities_Package.Append,
             File_ID    => File_ID,
             Aborted    => Aborted);

   ELSE
      File_Utilities_Package.
         Open_Output_File
            (Filename   => To_File,
             File_Mode  => File_Utilities_Package.Overwrite,
             File_ID    => File_ID,
             Aborted    => Aborted);

   END IF;

   IF NOT Aborted THEN                        -- file open/creation ok
      -- put all tag/filename pairs to output file
      FOR I in 1..Filename_List.Length LOOP
         File_Utilities_Package.
            Put_Token
               (Token_File => File_ID,
                Item       => Variable_String_Package.
                                 Text_Of(Temp_Ptr.File_Tag));
         File_Utilities_Package.
            Put_Token
               (Token_File => File_ID,
                Item       => Variable_String_Package.
                                 Text_Of(Temp_Ptr.Filename));
```

```
         -- blank line
         File_Utilities_Package.
            Put_Line(File_ID => File_ID, Item    => "");

         -- go to next pair
         Temp_Ptr := Temp_Ptr.Next;
      END LOOP;

      -- all done getting token pairs, close the file
      File_Utilities_Package.Close_Input_File(File_ID);

   ELSE
      RAISE Invalid_File;  -- problem w/ file open or creation

   END IF;

-- allow calling routine to handle the exceptions
EXCEPTION
   WHEN OTHERS =>
      RAISE;

END Save_Filename_List;



-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
--
-- This procedure is used to create a new list of filenames and tags.
-- "Generate_Filename_List", reads the filenames and the tags associated
-- with them from a file to build a filename list.  If the Action is
-- "Overwrite", the list will be overwritten with the new entries found in
-- the From_File.  If the Action is Append, the tag/filename pairs found
-- in the given From_File will be appended to the list.  The From_File
-- is checked for duplicate file tags and filenames and exceptions raised
-- when necessary.
--
-- Inputs:
--    Filename_List - tag/filename list to be manipulated
--    From_File     - external file from which to read the tag/filename
--                    pairs and put into the Filename_List
--    Action        - action to take on list, either append or overwrite
--
-- Outputs:
--    Filename_List - tag/filename list to be manipulated
--
-- Exceptions:
--    Invalid_File_Contents - raised when an expected token can not
--                            be found in the external file or
--                            if a token is too long ( > Max_Token_Length)
--    Invalid_File          - raised when there is a problem with the
--                            external file open or creation
-----------------------------------------------------------------------------
PROCEDURE Generate_Filename_List
             (Filename_List : IN OUT Filename_List_Type;
              From_File     : IN     STRING;
              Action        : IN     List_Action_Type) IS

   -- for opening file
   Successful : BOOLEAN                        := FALSE;

   -- file utilities package file identifier
```

```
   File_ID    : File_Utilities_Package.File_Identifier_Type;

   -- get_token length variables
   FN_Length  : NATURAL                        := 0;
   FT_Length  : NATURAL                        := 0;

   -- get_token tokens
   File_Tag   : STRING(1..Max_Token_Length) := (OTHERS => ' ');
   Filename   : STRING(1..Max_Token_Length) := (OTHERS => ' ');

   -- token strings are converted and stored in the list as vstrings
   VFile_Tag  : Variable_String_Package.VSTRING;
   VFilename  : Variable_String_Package.VSTRING;

   -- for counting the number of nodes in the list
   Node_Counter : NATURAL               := 1;

   -- temporary pointer used to manipulate tag/filename list
   Temp_Ptr   : File_Node_Access_Type          := Filename_List.Node_List;

BEGIN
   -- open the input file
   File_Utilities_Package.Open_Input_File
                            (Filename   => From_File,
                             File_ID    => File_ID,
                             Successful => Successful);

   IF Successful THEN  -- file opened ok

      -- clear list if Action is Overwrite
      IF Action = Overwrite THEN

         -- if current list not null, deallocate it
         IF Filename_List.Length > 0 THEN
            Clear_Filename_List(Filename_List);

         END IF;
      END IF;

      -- get all tag/filename pairs in input file
      WHILE NOT File_Utilities_Package.EOF(File_ID) LOOP

         -- get the file tag from the input file
         File_Utilities_Package.Get_Token
                                  (Token_File => File_ID,
                                   Token      => File_Tag,
                                   Length     => FT_Length);

         -- check to see that a token exists; if one doesn't, get out of
         -- the loop
         IF FT_Length = 0 THEN
            IF File_Utilities_Package.EOF(File_ID) THEN
               EXIT;

            ELSE
               RAISE Invalid_File_Contents;

            END IF;
         END IF;

         -- get the filename from the input file
         File_Utilities_Package.Get_Token
                                  (Token_File => File_ID,
```

```
                                        Token      => Filename,
                                        Length     => FN_Length);

            -- check to see that the corresponding filename token exists in
            -- the file; if one doesn't, raise the Invalid_File_Contents
            -- exception
            IF FN_Length = 0 THEN
               RAISE Invalid_File_Contents;

            ELSE
               Append_To_Filename_List
                  (Filename_List => Filename_List,
                   File_Tag      => File_Tag(1..FT_Length),
                   Filename      => Filename(1..FN_Length));

END LOOP;

         -- all done getting token pairs, close the file
         File_Utilities_Package.Close_Input_File(File_ID);

      ELSE
         RAISE Invalid_File;                -- problem w/ open

      END IF;

   -- allow calling routine to handle the exceptions
   EXCEPTION
      -- token too long
      WHEN Constraint_Error =>
         RAISE Invalid_File_Contents;

      WHEN OTHERS =>
         RAISE;

   END Generate_Filename_List;


   ------------------------------------------------------------------------------
   ------------------------------------------------------------------------------
   --
   -- This procedure allows the user to add a new file tag/filename pair to
   -- the end of the filename list.  Duplicate tags and filenames are not
   -- allowed and raise the Duplicate_Tag or Duplicate_Filename exceptions.
   --
   -- Inputs:
   --    Filename_List - tag/filename list to be add to
   --    File_Tag      - file tag to be added to list end
   --    Filename      - filename to be added to list end
   --
   -- Outputs:
   --    Filename_List - tag/filename list after addition of new pair
   --
   -- Exceptions:
   --    Duplicate_Filename - raised when the input filename is found to
   --                         already exist in the list
   --    Duplicate_File_Tag - raised when the input file tag is found to
   --                         already exist in the list
   ------------------------------------------------------------------------------
   PROCEDURE Append_To_Filename_List
                (Filename_List : IN OUT Filename_List_Type;
                 File_Tag      : IN     STRING;
                 Filename      : IN     STRING) IS
```

```
   -- token strings are converted and stored in the list as vstrings
   VFile_Tag    : Variable_String_Package.VSTRING;
   VFilename    : Variable_String_Package.VSTRING;

   -- for counting the number of nodes in the list
   Node_Counter : NATURAL               := 1;

   -- temporary pointer used to manipulate tag/filename list
   Temp_Ptr     : File_Node_Access_Type := Filename_List.Node_List;

BEGIN
   Node_Counter := 1;  -- initially

   -- first search list for duplicate tag & filenames
   IF Filename_List.Length > 0 THEN

      -- examine all nodes
      WHILE Node_Counter <= Filename_List.Length LOOP

         -- compare tags
         IF Convert_To_Upper(Variable_String_Package.
                                Text_Of(Temp_Ptr.File_Tag)) =
               Convert_To_Upper(File_Tag) THEN

            RAISE Duplicate_Tag;

         -- compare filenames
         ELSIF Convert_To_Upper(Variable_String_Package.
                                   Text_Of(Temp_Ptr.Filename)) =
                  Convert_To_Upper(Filename) THEN

            RAISE Duplicate_Filename;

         -- bump to next node
         ELSE
            Temp_Ptr := Temp_Ptr.Next;
            Node_Counter := Node_Counter + 1;

         END IF;
      END LOOP;
   END IF;

   -- convert the strings to vstrings
   Variable_String_Package.Make_Vstring (The_String  => File_Tag,
                                         New_Vstring => VFile_Tag);
   Variable_String_Package.Make_Vstring (The_String  => Filename,
                                         New_Vstring => VFilename);
   -- append a node to the list
   IF Filename_List.Length = 0 THEN
      -- for the first node, the temporary pointer should point
      -- to the new node and the filename list's node list should
      -- point to the first node
      Temp_Ptr := NEW File_Node_Type'(VFile_Tag, VFilename, NULL);
      Filename_List.Node_List := Temp_Ptr;

   ELSE
      -- reset the pointer to the front of the list
      Temp_Ptr := Filename_List.Node_List;

      -- find the end of the list if it is not empty and add new
      -- tag/filename pair there
      WHILE Temp_Ptr.Next /= NULL LOOP
         Temp_Ptr := Temp_Ptr.Next;
```

```
      END LOOP;

      -- append new node
      Temp_Ptr.Next := NEW File_Node_Type'(VFile_Tag, VFilename, NULL);
   END IF;

   -- increment the list length
   Filename_List.Length := Filename_List.Length + 1;

-- allow calling routine to handle the exceptions
EXCEPTION
   WHEN OTHERS =>
      RAISE;

END Append_To_Filename_List;



-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
--
-- This procedure allows the user to do a "sorted" insert of a new
-- file tag/filename pair into the filename list.  The pair will be
-- alphanumerically inserted either by file tag or by filename.  The
-- routine assumes that the list is already properly sorted for the
-- desired insert.  Results will be unpredictable if either the list
-- is not already sorted or if the pair is inserted by the wrong
-- token (tag or name).  Duplicate tags and filenames are not
-- allowed and raise the Duplicate_Tag or Duplicate_Filename exceptions.
--
-- Inputs:
--    Filename_List - tag/filename list to be add to
--    File_Tag      - file tag to be added to list end
--    Filename      - filename to be added to list end
--    Insert        - insert type (by file tag or filename)
--
-- Outputs:
--    Filename_List - tag/filename list after addition of new pair
--
-- Exceptions:
--    Duplicate_Filename - raised when the input filename is found to
--                         already exist in the list
--    Duplicate_File_Tag - raised when the input file tag is found to
--                         already exist in the list
-------------------------------------------------------------------------------
PROCEDURE Insert_Into_Filename_List
             (Filename_List : IN OUT Filename_List_Type;
              File_Tag      : IN     STRING;
              Filename      : IN     STRING;
              Insert        : IN     List_Sort_Type) IS


   -- token strings are converted and stored in the list as vstrings
   VFile_Tag    : Variable_String_Package.VSTRING;
   VFilename    : Variable_String_Package.VSTRING;

   -- for counting the number of nodes in the list
   Node_Counter : NATURAL               := 1;

   -- temporary pointers used to manipulate tag/filename list
   Temp_Ptr     : File_Node_Access_Type := Filename_List.Node_List;

   -- local boolean
```

```
   Insert_Made  : BOOLEAN               := FALSE;


BEGIN
   Node_Counter := 1;  -- initially

   -- first search list for duplicate tag & filenames
   IF Filename_List.Length > 0 THEN

      -- examine all nodes
      WHILE Node_Counter <= Filename_List.Length LOOP

         -- compare tags
         IF Convert_To_Upper(Variable_String_Package.
                              Text_Of(Temp_Ptr.File_Tag)) =
               Convert_To_Upper(File_Tag) THEN

            RAISE Duplicate_Tag;

         -- compare filenames
         ELSIF Convert_To_Upper(Variable_String_Package.
                                 Text_Of(Temp_Ptr.Filename)) =
                  Convert_To_Upper(Filename) THEN

            RAISE Duplicate_Filename;

         -- bump to next node
         ELSE
            Temp_Ptr := Temp_Ptr.Next;
            Node_Counter := Node_Counter + 1;

         END IF;
      END LOOP;
   END IF;

   -- reset the pointers to the front of the list
   Temp_Ptr := Filename_List.Node_List;

   -- convert the strings to vstrings
   Variable_String_Package.Make_Vstring (The_String  => File_Tag,
                                          New_Vstring => VFile_Tag);
   Variable_String_Package.Make_Vstring (The_String  => Filename,
                                          New_Vstring => VFilename);
   -- insert a node into the list
   IF Filename_List.Length = 0 THEN

      -- for the first node, the temporary pointer should point
      -- to the new node and the filename list's node list should
      -- point to the first node
      Temp_Ptr := NEW File_Node_Type'(VFile_Tag, VFilename, NULL);
      Filename_List.Node_List := Temp_Ptr;

   ELSE
      -- prepend new node
      IF (Insert = By_File_Tag AND

          Convert_To_Upper(File_Tag) <
             Convert_To_Upper(Variable_String_Package.
                               Text_Of(Temp_Ptr.File_Tag))) OR

         (Insert = By_Filename AND

          Convert_To_Upper(Filename) <
```

```
             Convert_To_Upper(Variable_String_Package.
                                 Text_Of(Temp_Ptr.Filename))) THEN

            Temp_Ptr := NEW File_Node_Type'
                            (VFile_Tag, VFilename, Filename_List.Node_List);
            Filename_List.Node_List := Temp_Ptr;

      ELSE
         Insert_Made := FALSE;

         -- find the node to insert at if the list is not empty and add new
         -- tag/filename pair there
         WHILE Temp_Ptr.Next /= NULL LOOP

            IF (Insert = By_File_Tag AND
                 Convert_To_Upper(File_Tag) <
                  Convert_To_Upper(Variable_String_Package.
                                  Text_Of(Temp_Ptr.Next.File_Tag))) OR

               (Insert = By_Filename AND
                 Convert_To_Upper(Filename) <
                  Convert_To_Upper(Variable_String_Package.
                                  Text_Of(Temp_Ptr.Next.Filename))) THEN

                  Temp_Ptr.Next := NEW File_Node_Type'
                                    (VFile_Tag, VFilename, Temp_Ptr.Next);
                  Insert_Made := TRUE;
                  EXIT;

            ELSE
               Temp_Ptr := Temp_Ptr.Next;

            END IF;
         END LOOP;

         IF NOT Insert_Made THEN
            -- append new node
            Temp_Ptr.Next := NEW File_Node_Type'
                                (VFile_Tag, VFilename, NULL);
         END IF;
      END IF;
   END IF;

   -- increment the list length
   Filename_List.Length := Filename_List.Length + 1;

-- allow calling routine to handle the exceptions
EXCEPTION
   WHEN OTHERS =>
      RAISE;

END Insert_Into_Filename_List;



-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
--
-- This procedure allows the user to delete a file tag/filename pair
-- from a filename list.  The list is searched for the given file tag and
-- when found, the tag/filename pair are deleted.  The Invalid_Tag
-- exception will be raised if the file tag is not found.
--
```

```
-- Inputs:
--    Filename_List - tag/filename list to be deleted from
--    File_Tag      - file tag to delete (filename deleted also)
--
-- Outputs:
--    Filename_List - tag/filename list after addition of new pair
--
-- Exceptions:
--    Invalid_Tag - raised when the input file tag is not found in the list
-------------------------------------------------------------------------------
PROCEDURE Delete_File_Tag
             (Filename_List : IN OUT Filename_List_Type;
              File_Tag      : IN     STRING) IS

   -- for counting the number of nodes in the list
   Node_Counter : NATURAL               := 1;

   -- temporary pointers used to manipulate tag/filename list
   Temp_Ptr     : File_Node_Access_Type := Filename_List.Node_List;
   Prev_Ptr     : File_Node_Access_Type := Filename_List.Node_List;

   -- boolean used to control search
   Found        : BOOLEAN               := FALSE;

BEGIN

   -- initialize
   Found := FALSE;
   Node_Counter := 1;

   -- examine every node until found or end of list is reached
   WHILE Node_Counter <= Filename_List.Length AND NOT Found LOOP

      -- comparisons are not case sensitive; convert both to upper case
      -- and then compare them
      IF Convert_To_Upper
            (Variable_String_Package.
                Text_Of(Temp_Ptr.File_Tag)) = Convert_To_Upper(File_Tag) THEN

         Found := TRUE;

         -- first node
         IF Node_Counter = 1 THEN
            Filename_List.Node_List := Temp_Ptr.Next; -- move to next node

         -- last node
         ELSIF Node_Counter = Filename_List.Length THEN
            -- bump to next to last node
            FOR I IN 1..Filename_List.Length - 2 LOOP
               Prev_Ptr := Prev_Ptr.Next;
            END LOOP;

            Prev_Ptr.Next := NULL;  -- set new last node to null

         -- all others
         ELSE
            -- bump to next to node just before the one to be deleted
            FOR I IN 1..Node_Counter - 2 LOOP
               Prev_Ptr := Prev_Ptr.Next;
            END LOOP;

            Prev_Ptr.Next := Temp_Ptr.Next;  -- circumvent deleted node
```

```
         END IF;
         Dispose (Temp_Ptr); -- dispose of deleted node
         Filename_List.Length := Filename_List.Length - 1;

      -- bump to next node; increment node counter
      ELSE
         Temp_Ptr := Temp_Ptr.Next;
         Node_Counter := Node_Counter + 1;
      END IF;
   END LOOP;

   -- done searching list; if the file tag was not found raise the
   -- Invalid_Tag exception
   IF NOT Found THEN
      RAISE Invalid_Tag;
   END IF;

-- allow calling routine to handle the exceptions
EXCEPTION
   WHEN OTHERS =>
      RAISE;

END Delete_File_Tag;


-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
--
-- This procedure allows the user to delete a file tag/filename pair
-- from a filename list.  The list is searched for the given filename and
-- when found, the tag/filename pair are deleted.  The Invalid_Filename
-- exception will be raised if the filename is not found.
--
-- Inputs:
--    Filename_List - tag/filename list to be deleted from
--    Filename      - filename to delete (file tag deleted also)
--
-- Outputs:
--    Filename_List - tag/filename list after addition of new pair
--
-- Exceptions:
--    Invalid_Filename - raised when the input filename is not found
--                       in the list
-----------------------------------------------------------------------------
PROCEDURE Delete_Filename
             (Filename_List : IN OUT Filename_List_Type;
              Filename      : IN     STRING) IS

   -- for counting the number of nodes in the list
   Node_Counter : NATURAL               := 1;

   -- temporary pointers used to manipulate tag/filename list
   Temp_Ptr     : File_Node_Access_Type := Filename_List.Node_List;
   Prev_Ptr     : File_Node_Access_Type := Filename_List.Node_List;

   -- boolean used to control search
   Found        : BOOLEAN               := FALSE;

BEGIN
   -- initialize
   Found := FALSE;
   Node_Counter := 1;
```

```
   -- examine every node until found or end of list is reached
   WHILE Node_Counter <= Filename_List.Length AND NOT Found LOOP

      -- comparisons are not case sensitive; convert both to upper case
      -- and then compare them
      IF Convert_To_Upper
            (Variable_String_Package.
                Text_Of(Temp_Ptr.Filename)) = Convert_To_Upper(Filename) THEN

         Found := TRUE;

         -- first node
         IF Node_Counter = 1 THEN
            Filename_List.Node_List := Temp_Ptr.Next; -- move to next node

         -- last node
         ELSIF Node_Counter = Filename_List.Length THEN
            -- bump to next to last node
            FOR I IN 1..Filename_List.Length - 2 LOOP
               Prev_Ptr := Prev_Ptr.Next;
            END LOOP;

            Prev_Ptr.Next := NULL;  -- set new last node to null

         -- all others
         ELSE
            -- bump to next to node just before the one to be deleted
            FOR I IN 1..Node_Counter - 2 LOOP
               Prev_Ptr := Prev_Ptr.Next;
            END LOOP;

            Prev_Ptr.Next := Temp_Ptr.Next;  -- circumvent deleted node

         END IF;
         Dispose (Temp_Ptr); -- dispose of deleted node
         Filename_List.Length := Filename_List.Length - 1;

      -- bump to next node; increment node counter
      ELSE
         Temp_Ptr := Temp_Ptr.Next;
         Node_Counter := Node_Counter + 1;
      END IF;
   END LOOP;

   -- done searching list; if the filename was not found raise the
   -- Invalid_Filename exception
   IF NOT Found THEN
      RAISE Invalid_Filename;
   END IF;

-- allow calling routine to handle the exceptions
EXCEPTION
   WHEN OTHERS =>
      RAISE;

END Delete_Filename;



-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
--
-- This function searches a filename list for a filename and returns
```

```
-- a boolean telling whether or not it was found.
--
-- Inputs:
--    Filename_List - tag/filename list to be searched
--    Filename      - filename to search for in list
--
-- Outputs:
--
-- Exceptions:
-------------------------------------------------------------------------------
FUNCTION Filename_Exists
            (Filename_List : IN     Filename_List_Type;
             Filename      : IN     STRING)  RETURN BOOLEAN IS

   -- for counting the number of nodes in the list
   Node_Counter : NATURAL               := 1;

   -- temporary pointer used to manipulate tag/filename list
   Temp_Ptr     : File_Node_Access_Type := Filename_List.Node_List;

   -- boolean used to control search
   Found        : BOOLEAN               := FALSE;

BEGIN

   -- initialize
   Found := FALSE;
   Node_Counter := 1;

   -- examine every node until found or end of list is reached
   WHILE Node_Counter <= Filename_List.Length AND NOT Found LOOP

      -- comparisons are not case sensitive; convert both to upper case
      -- and then compare them
      IF Convert_To_Upper
            (Variable_String_Package.
                Text_Of(Temp_Ptr.Filename)) = Convert_To_Upper(Filename) THEN

         Found := TRUE;  -- search is done, exit loop

      -- bump to next node; increment node counter
      ELSE
         Temp_Ptr := Temp_Ptr.Next;
         Node_Counter := Node_Counter + 1;
      END IF;
   END LOOP;

   RETURN Found;

-- allow calling routine to handle the exceptions
EXCEPTION
   WHEN OTHERS =>
      RAISE;

END Filename_Exists;



-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
--
-- This function searches a filename list for a file tag and returns
```

```
-- a boolean telling whether or not it was found.
--
-- Inputs:
--    Filename_List - tag/filename list to be searched
--    File_Tag      - tag to search for in list
--
-- Outputs:
--
-- Exceptions:
-------------------------------------------------------------------------------
FUNCTION File_Tag_Exists
            (Filename_List : IN     Filename_List_Type;
             File_Tag      : IN     STRING)  RETURN BOOLEAN IS

   -- for counting the number of nodes in the list
   Node_Counter : NATURAL               := 1;

   -- temporary pointer used to manipulate tag/filename list
   Temp_Ptr     : File_Node_Access_Type := Filename_List.Node_List;

   -- boolean used to control search
   Found        : BOOLEAN               := FALSE;

BEGIN

   -- initialize
   Found := FALSE;
   Node_Counter := 1;

   -- examine every node until found or end of list is reached
   WHILE Node_Counter <= Filename_List.Length AND NOT Found LOOP

      -- comparisons are not case sensitive; convert both to upper case
      -- and then compare them
      IF Convert_To_Upper
            (Variable_String_Package.
                Text_Of(Temp_Ptr.File_Tag)) = Convert_To_Upper(File_Tag) THEN

         Found := TRUE;  -- search is done, exit loop

      -- bump to next node; increment node counter
      ELSE
         Temp_Ptr := Temp_Ptr.Next;
         Node_Counter := Node_Counter + 1;
      END IF;
   END LOOP;

   RETURN Found;

-- allow calling routine to handle the exceptions
EXCEPTION
   WHEN OTHERS =>
      RAISE;

END File_Tag_Exists;



-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
--
-- This function searches the specified list for the file tag passed
```

```
-- to it and returns the file name associated with that tag.  The
-- tag comparisons are NOT case sensitive.
--
-- Inputs:
--    Filename_List - tag/filename list to be searched
--    File_Tag      - tag to search for in list
--
-- Outputs:
--
-- Exceptions:
--    Invalid_Tag   - raised when the input file tag is not found
--                    in the list
-----------------------------------------------------------------------------
FUNCTION Get_Filename
            (Filename_List : IN     Filename_List_Type;
             File_Tag      : IN     STRING)  RETURN STRING IS

   -- for counting the number of nodes in the list
   Node_Counter : NATURAL               := 1;

   -- temporary pointer used to manipulate tag/filename list
   Temp_Ptr     : File_Node_Access_Type := Filename_List.Node_List;

   -- boolean used to control search
   Found        : BOOLEAN               := FALSE;

BEGIN
   -- initialize
   Found := FALSE;
   Node_Counter := 1;

   -- examine every node until found or end of list is reached
   WHILE Node_Counter <= Filename_List.Length AND NOT Found LOOP

      -- comparisons are not case sensitive; convert both to upper case
      -- and then compare them
      IF Convert_To_Upper
            (Variable_String_Package.
                Text_Of(Temp_Ptr.File_Tag)) = Convert_To_Upper(File_Tag) THEN

         Found := TRUE;  -- search is done, exit loop

      -- bump to next node; increment node counter
      ELSE
         Temp_Ptr := Temp_Ptr.Next;
         Node_Counter := Node_Counter + 1;
      END IF;
   END LOOP;

   -- done searching list; if the tag was found return the filename;
   -- otherwise, raise the Invalid_Tag exception
   IF Found THEN
      RETURN Variable_String_Package.Text_Of(Temp_Ptr.Filename);

   ELSE
      RAISE Invalid_Tag;

   END IF;

-- allow calling routine to handle the exceptions
EXCEPTION
   WHEN OTHERS =>
      RAISE;
```

```
END Get_Filename;


-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
--
-- This function searches the specified list for the filename passed
-- to it and returns the file tag associated with that filename.  The
-- filename comparisons are NOT case sensitive.
--
-- Inputs:
--    Filename_List - tag/filename list to be searched
--    Filename      - filename to search for in list
--
-- Outputs:
--
-- Exceptions:
--    Invalid_Filename - raised when the input filename is not found
--                       in the list
-----------------------------------------------------------------------------
FUNCTION Get_File_Tag
            (Filename_List : IN     Filename_List_Type;
             Filename      : IN     STRING)  RETURN STRING IS

   -- for counting the number of nodes in the list
   Node_Counter : NATURAL               := 1;

   -- temporary pointer used to manipulate tag/filename list
   Temp_Ptr     : File_Node_Access_Type := Filename_List.Node_List;

   -- boolean used to control search
   Found        : BOOLEAN               := FALSE;

BEGIN
   -- initialize
   Found := FALSE;
   Node_Counter := 1;

   -- examine every node until found or end of list is reached
   WHILE Node_Counter <= Filename_List.Length AND NOT Found LOOP

      -- comparisons are not case sensitive; convert both to upper case
      -- and then compare them
      IF Convert_To_Upper
            (Variable_String_Package.
                Text_Of(Temp_Ptr.Filename)) = Convert_To_Upper(Filename) THEN

         Found := TRUE;  -- search is done, exit loop

      -- bump to next node; increment node counter
      ELSE
         Temp_Ptr := Temp_Ptr.Next;
         Node_Counter := Node_Counter + 1;

      END IF;
   END LOOP;

   -- done searching list; if the filename was found return the file tag;
   -- otherwise, raise the Invalid_Filename exception
   IF Found THEN
      RETURN Variable_String_Package.Text_Of(Temp_Ptr.File_Tag);
```

```
   ELSE
      RAISE Invalid_Filename;

   END IF;

-- allow calling routine to handle the exceptions
EXCEPTION
   WHEN OTHERS =>
      RAISE;

END Get_File_Tag;


------------------------------------------------------------------------------
------------------------------------------------------------------------------
--
-- This function finds the node specified by the position passed
-- in and returns the filename at that position.
--
-- Inputs:
--    Filename_List - tag/filename list to be searched
--    Position      - position in list to return filename from
--
-- Outputs:
--
-- Exceptions:
--    Invalid_Position - position does not exist for input list
------------------------------------------------------------------------------
FUNCTION Get_Filename
            (Filename_List : IN     Filename_List_Type;
             Position      : IN     Natural)  RETURN STRING IS

   -- temporary pointer used to manipulate tag/filename list
   Temp_Ptr : File_Node_Access_Type := Filename_List.Node_List;

BEGIN
   -- make sure that the postion is valid; if not, raise exception
   IF Position = 0 OR Position > Filename_List.Length THEN
      RAISE Invalid_Position;

   ELSE
      -- traverse list to desired position
      FOR I IN 1..Position - 1 LOOP
         Temp_Ptr := Temp_Ptr.Next;
      END LOOP;

      -- return the filename at the desired position
      RETURN Variable_String_Package.Text_Of(Temp_Ptr.Filename);

   END IF;

-- allow calling routine to handle the exceptions
EXCEPTION
   WHEN OTHERS =>
      RAISE;

END Get_Filename;


------------------------------------------------------------------------------
------------------------------------------------------------------------------
--
```

```
-- This function finds the node specified by the position passed
-- in and returns the file tag at that position.
--
-- Inputs:
--    Filename_List - tag/filename list to be searched
--    Position      - position in list to return file tag from
--
-- Outputs:
--
-- Exceptions:
--    Invalid_Position - position does not exist for input list
-------------------------------------------------------------------------------
FUNCTION Get_File_Tag
            (Filename_List : IN     Filename_List_Type;
             Position      : IN     Natural)  RETURN STRING IS

   -- temporary pointer used to manipulate tag/filename list
   Temp_Ptr : File_Node_Access_Type := Filename_List.Node_List;

BEGIN
   -- make sure that the postion is valid; if not, raise exception
   IF Position = 0 OR Position > Filename_List.Length THEN
      RAISE Invalid_Position;

   ELSE
      -- traverse list to desired position
      FOR I IN 1..Position - 1 LOOP
         Temp_Ptr := Temp_Ptr.Next;
      END LOOP;

      -- return the file tag at the desired position
      RETURN Variable_String_Package.Text_Of(Temp_Ptr.File_Tag);

   END IF;

-- allow calling routine to handle the exceptions
EXCEPTION
   WHEN OTHERS =>
      RAISE;

END Get_File_Tag;


-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
--
-- This function searches for the specified filename in the list, and
-- returns the position in the list of the node which contains it.
--
-- Inputs:
--    Filename_List - tag/filename list to be searched
--    Filename      - filename to search for in list
--
-- Outputs:
--
-- Exceptions:
--    Invalid_Filename - raised when the input filename is not in the
--                       input list
-------------------------------------------------------------------------------
FUNCTION Get_Filename_Position
            (Filename_List : IN     Filename_List_Type;
             Filename      : IN     STRING)  RETURN NATURAL IS
```

```
   -- for counting the number of nodes in the list and returning
   -- the position at which the input filename is found
   Position : NATURAL := 1;

   -- temporary pointer used to manipulate tag/filename list
   Temp_Ptr     : File_Node_Access_Type := Filename_List.Node_List;

   -- boolean used to control search
   Found        : BOOLEAN               := FALSE;

BEGIN
   -- initialize
   Found := FALSE;
   Position := 1;

   -- examine every node until found or end of list is reached
   WHILE Position <= Filename_List.Length AND NOT Found LOOP

      -- comparisons are not case sensitive; convert both to upper case
      -- and then compare them
      IF Convert_To_Upper
            (Variable_String_Package.
                Text_Of(Temp_Ptr.Filename)) = Convert_To_Upper(Filename) THEN

         Found := TRUE;  -- search is done, exit loop

      -- bump to next node; increment node counter
      ELSE
         Temp_Ptr := Temp_Ptr.Next;
         Position := Position + 1;

      END IF;
   END LOOP;

   -- done searching list; if the filename was found return the position;
   -- otherwise, raise the Invalid_Filename exception
   IF Found THEN
      RETURN Position;

   ELSE
      RAISE Invalid_Filename;

   END IF;

-- allow calling routine to handle the exceptions
EXCEPTION
   WHEN OTHERS =>
      RAISE;

END Get_Filename_Position;


------------------------------------------------------------------------------
------------------------------------------------------------------------------
--
-- This function searches for the specified file tag in the list, and
-- returns the position in the list of the node which contains it.
--
-- Inputs:
--    Filename_List - tag/filename list to be searched
--    File_Tag      - file tag to search for in list
--
-- Outputs:
```

```
--
-- Exceptions:
--    Invalid_File_Tag - raised when the input file tag is not in the
--                       input list
-----------------------------------------------------------------------------
FUNCTION Get_File_Tag_Position
            (Filename_List : IN     Filename_List_Type;
             File_Tag      : IN     STRING)  RETURN NATURAL IS

   -- for counting the number of nodes in the list and returning
   -- the position at which the input file tag is found
   Position : NATURAL := 1;

   -- temporary pointer used to manipulate tag/filename list
   Temp_Ptr     : File_Node_Access_Type := Filename_List.Node_List;

   -- boolean used to control search
   Found        : BOOLEAN               := FALSE;

BEGIN
   -- initialize
   Found := FALSE;
   Position := 1;

   -- examine every node until found or end of list is reached
   WHILE Position <= Filename_List.Length AND NOT Found LOOP

      -- comparisons are not case sensitive; convert both to upper case
      -- and then compare them
      IF Convert_To_Upper
            (Variable_String_Package.
                Text_Of(Temp_Ptr.File_Tag)) = Convert_To_Upper(File_Tag) THEN

         Found := TRUE;  -- search is done, exit loop

      -- bump to next node; increment node counter
      ELSE
         Temp_Ptr := Temp_Ptr.Next;
         Position := Position + 1;

      END IF;
   END LOOP;

   -- done searching list; if the file tag was found return the position;
   -- otherwise, raise the Invalid_File_Tag exception
   IF Found THEN
      RETURN Position;

   ELSE
      RAISE Invalid_Tag;

   END IF;

-- allow calling routine to handle the exceptions
EXCEPTION
   WHEN OTHERS =>
      RAISE;

END Get_File_Tag_Position;



-----------------------------------------------------------------------------
```

```
-----------------------------------------------------------------------------
--
-- This procedure sorts a list.  The user has the option to do the
-- reorder by file tag or filename.  The procedure uses a bubble sort
-- swapping pointers when necessary instead of exchanging file tags and
-- filenames.
--
-- Inputs:
--    Filename_List - tag/filename list to be sorted
--    Sort          - sort type (by file tag or filename
--
-- Outputs:
--    Filename_List - tag/filename list to be sorted
--
-- Exceptions:
-----------------------------------------------------------------------------
PROCEDURE Sort_Filename_List
             (Filename_List : IN OUT Filename_List_Type;
              Sort          : IN     List_Sort_Type) IS

   -- temporary pointers used to manipulate tag/filename list
   Temp_Ptr     : File_Node_Access_Type := Filename_List.Node_List;
   Prev_Ptr     : File_Node_Access_Type := NULL;

   -- local booleans
   No_Exchange  : BOOLEAN               := TRUE;
   Swap         : BOOLEAN               := FALSE;


BEGIN

   LOOP

      -- initially point to front of list on each pass of this outer loop
      Temp_Ptr := Filename_List.Node_List;
      Prev_Ptr := NULL;

      -- initially
      No_Exchange := TRUE;

      WHILE Temp_Ptr.Next /= NULL LOOP

         -- check sort type and compare tag or filename at current node
         -- and next node.  Swap if necessary.
         Swap := FALSE;
         IF Sort = By_File_Tag THEN

            IF Convert_To_Upper
                  (Variable_String_Package
                     .Text_Of(Temp_Ptr.File_Tag)) >
               Convert_To_Upper
                  (Variable_String_Package.
                     Text_Of(Temp_Ptr.Next.File_Tag)) THEN

               Swap := TRUE;

            END IF;

         ELSE

            IF Convert_To_Upper
                  (Variable_String_Package.
                     Text_Of(Temp_Ptr.Filename)) >
```

```
                  Convert_To_Upper
                     (Variable_String_Package.
                        Text_Of(Temp_Ptr.Next.Filename)) THEN

                  Swap := TRUE;

               END IF;

            END IF;

            -- if swap is necessary, just swap pointers
            IF Swap THEN

               -- node to swap is first node
               IF Temp_Ptr = Filename_List.Node_List THEN
                  Filename_List.Node_List := Temp_Ptr.Next;
                  Prev_Ptr := Filename_List.Node_List;
                  Temp_Ptr.Next := Temp_Ptr.Next.Next;
                  Prev_Ptr.Next := Temp_Ptr;

               -- all other nodes
               ELSE
                  Prev_Ptr.Next := Temp_Ptr.Next;
                  Temp_Ptr.Next := Temp_Ptr.Next.Next;
                  Prev_Ptr.Next.Next := Temp_Ptr;
                  Prev_Ptr := Prev_Ptr.Next;

               END IF;

               -- made a swap
               No_Exchange := FALSE;

            ELSE
               -- no swap made; go to next node
               Prev_Ptr := Temp_Ptr;
               Temp_Ptr := Temp_Ptr.Next;

            END IF;

         END LOOP;

         -- list is sorted
         EXIT WHEN No_Exchange;

      END LOOP;

   -- allow calling routine to handle the exceptions
   EXCEPTION
      WHEN OTHERS =>
         RAISE;

   END Sort_Filename_List;

   ------------------------------------------------------------------------------
   ------------------------------------------------------------------------------

END Filename_Manager_Package;
```

Appendix 3: Constants Package

```
-------------------------------------------------------------------------------
-- File Name:       CFGCONST.ADS (spec only version)
-- Organization:    NASA - Langley Research Center (LaRC)
-- Project:         LIDAR In-Space Technology Experiment (LITE)
-------------------------------------------------------------------------------
--*****************************************************************************
-- Name/Number:
--    System_Constants               (package spec)
--
-- Abstract:
--    This package spec contains several constants which are used in the
--    CFG User Interface software to define color schemes, file name
--    extensions and directories, Screen mode constants, and others.  It
--    has no body.
--
-- Acronyms/Abbreviations:
--    None
--
-- Dependencies:
--    Menu_Package
--    Screen_Package
--    Keyboard_Package
--
-- Global Objects:
--    All
--
-- Exceptions:
--    None
--
-- Machine/Compiler Dependencies:
--
--*****************************************************************************

with Unsigned;
with Menu_Package;
with Screen_Package;
with Keyboard_Package;



Package System_Constants Is



   -- This is the main color scheme used in the menus and windows of the
   -- user interface on the Master AT.
   Normal_Color_Scheme : constant Menu_Package.Color_Scheme_Type :=
      (Screen_Package.Blue_Background,         -- Background_Color
       Screen_Package.Bright_White_Foreground, -- Foreround_Color
       Screen_Package.Yellow_Foreground,       -- Title_Color
       Screen_Package.Yellow_Foreground,       -- Border_Color
       Screen_Package.Light_Red_Foreground);   -- Highlight_Color


   -- This color scheme is used on the Master AT when a menu or window is
   -- "grayed out" after the user has made a selection.
   Inactive_Color_Scheme : constant Menu_Package.Color_Scheme_Type :=
      (Screen_Package.Blue_Background,         -- Background_Color
       Screen_Package.White_Foreground,        -- Foreround_Color
       Screen_Package.Yellow_Foreground,       -- Title_Color
```

```
    Screen_Package.White_Foreground,        -- Border_Color
    Screen_Package.Black_Foreground);       -- Highlight_Color

Warning_CScheme : CONSTANT Menu_Package.Color_Scheme_Type :=
   (Screen_Package.Red_Background,          -- Background_Color
    Screen_Package.Bright_White_Foreground, -- Foreround_Color
    Screen_Package.Yellow_Foreground,       -- Title_Color
    Screen_Package.Yellow_Foreground,       -- Border_Color
    Screen_Package.Yellow_Foreground);      -- Highlight_Color

Standby_Warning_CScheme : CONSTANT Menu_Package.Color_Scheme_Type :=
   (Screen_Package.Magenta_Background,      -- Background_Color
    Screen_Package.Light_Green_Foreground,  -- Foreround_Color
    Screen_Package.Bright_White_Foreground, -- Title_Color
    Screen_Package.Yellow_Foreground,       -- Border_Color
    Screen_Package.Yellow_Foreground);      -- Highlight_Color

-- These constants define the size of the screen for anybody that needs
-- to know these things
Screen_Mode       : CONSTANT Screen_Package.Text_Mode_Type := Screen_Package.Color_80x50;
Number_of_Columns : CONSTANT Screen_Package.Column_Type    := 80;
Number_Of_Rows    : CONSTANT Screen_Package.Row_Type       := 50;

File_Name_Length      : CONSTANT := 8;
Full_File_Name_Length : CONSTANT := 12; -- this includes the file extension

-- 600 bytes in NASCOM block --
NASCOM_Block_Length             : CONSTANT integer := 600;

Bit_Size                        : CONSTANT := 1;
Nibble_Size                     : CONSTANT := 4;
Word_Size                       : CONSTANT := 16;

Lookup_Table_1_Minimum_Pressure : CONSTANT FLOAT := 15.0;
Lookup_Table_1_Maximum_Pressure : CONSTANT FLOAT := 65.0;
Lookup_Table_2_Minimum_Pressure : CONSTANT FLOAT :=  5.0;
Lookup_Table_2_Maximum_Pressure : CONSTANT FLOAT := 65.0;

-- maximum number of command words a command block can contain
Max_Command_Words               : CONSTANT := 28;

-- maximum number of command words in a command block transferred to
-- the IC via the PGSC
Max_Via_PGSC_Command_Words      : CONSTANT := 22;

-- maximum number of words in a single command block, including header
-- information words
Max_Block_Size                  : CONSTANT := 30;

S_Key_Array     : CONSTANT Keyboard_Package.Special_Keys_Array_Type (1..3) :=
                  (Keyboard_Package.F1_Key, Keyboard_Package.F2_Key,
                   Keyboard_Package.F10_Key);
IC_S_Key_Array  : CONSTANT Keyboard_Package.Special_Keys_Array_Type (1..4) :=
                  (Keyboard_Package.F1_Key, Keyboard_Package.F2_Key,
                   Keyboard_Package.F3_Key, Keyboard_Package.F10_Key);
No_F1_Key_Array : CONSTANT Keyboard_Package.Special_Keys_Array_Type (1..2) :=
                  (Keyboard_Package.F2_Key, Keyboard_Package.F10_Key);
F10_Key_Array   : CONSTANT Keyboard_Package.Special_Keys_Array_Type (1..1) :=
                  (OTHERS => Keyboard_Package.F10_Key);

Extra_Chars     : CONSTANT STRING           := " .,<>?/\';:[]{}-_=+|`~!@#$%^&*()";
Blanks       : CONSTANT STRING (1 .. 80) := (OTHERS => ' ');
```

```
-- This is the maximum number of Time Specific Commands that could be in
-- a single LITE command block (if all commands were non-parametric,
-- time-specific commands, they would be 3 words each, and so at most 9 of
-- them could fit in a 28-word LITE command block).
Max_TS_Cmds_Per_Block      : CONSTANT := 9;
Max_TS_Commands            : CONSTANT := 32;

Command_Menu_Width         : CONSTANT := 24;



-- this offset is added to the hot key position read in from the data file, so that
-- all mnemonics in the menu are a few spaces over from the left edge of the menu
Menu_Offset                : CONSTANT := 3;
Menu_Offset_String         : CONSTANT STRING (1..3) := (OTHERS => ' ');

Max_Mnemonic_Length        : CONSTANT := 15;
Descriptor_Length          : CONSTANT := 40;
Units_String_Max           : CONSTANT := 5;

-- Command Code constants for uplink code mode commands
Byte_IO_Write_Command_Code : CONSTANT := 16#50#;
Word_IO_Write_Command_Code : CONSTANT := 16#51#;
Memory_Write_Command_Code  : CONSTANT := 16#54#;
Memory_Fill_Command_Code   : CONSTANT := 16#56#;


-- LITE IC null commands
LITE_IC_Null_Cmd_Fill_Pattern_High_Byte : CONSTANT := 16#04#;
LITE_IC_Null_Cmd_Fill_Pattern_Low_Byte  : CONSTANT := 16#FF#;

Reserved                : CONSTANT Unsigned.Byte := 0;
-- row offset for next menu after drawing an inactive box around the
-- previous menu selection
Show_Menu_Choice_Offset : CONSTANT := 5;

-- there are 28 Memory read strings that can be displayed at one time; this
-- area of the display is broken down into 28 rows of 45 character strings
Max_Memory_Reads        : CONSTANT POSITIVE := 28; -- up to 28 mem read displayable
Memory_String_Length    : CONSTANT          := 45; -- string length


-- there are 45 IO read strings that can be displayed at one time; this area
-- of the display is broken down into 9 rows and 5 columns of 9 character
-- strings
Max_IO_Rows         : CONSTANT POSITIVE := 9;  -- max IO read rows
Max_IO_Cols         : CONSTANT POSITIVE := 5;  -- max IO read cols
IO_String_Length    : CONSTANT POSITIVE := 9;  -- individual read string length
Max_IO_Reads        : CONSTANT POSITIVE := Max_IO_Rows * Max_IO_Cols; -- max mem reads

IC_Command_Frame          : CONSTANT := 8;
IC_Memory_Frame           : CONSTANT := 10;
IC_Uplink_Byte            : CONSTANT := 53;

Byte_IO_Read_Command_Code : CONSTANT := 16#52#;
Word_IO_Read_Command_Code : CONSTANT := 16#53#;
Memory_Read_Command_Code  : CONSTANT := 16#55#;

PDI_Sync_Pattern          : CONSTANT Unsigned.Word := 16#BEEF#;

-- min and max screen rows
Max_Top_Row          : CONSTANT Screen_Package.Row_Type := 1;
```

```
Max_Bottom_Row       : CONSTANT Screen_Package.Row_Type := 49;
Max_Column           : CONSTANT                         := 79;

-- number of rows in the general error window
Rows_In_Error_Window : CONSTANT := 5;

-- user input and title row offsets from top of many prompt windows
Input_Offset_Row     : CONSTANT Screen_Package.Row_Type := 2;
Title_Offset_Row     : CONSTANT Screen_Package.Row_Type := 2;

-- bottom row offset from top for many prompt windows
Bottom_Row_Offset    : CONSTANT Screen_Package.Row_Type := 4;

-- # of rows in a yes/no box
Yes_No_Box_Rows      : CONSTANT Screen_Package.Row_Type := 6;

Seconds_Per_Day      : CONSTANT Integer := 86400;
Seconds_Per_Hour     : CONSTANT Integer := 3600;

-- These constants define the "boiler plate" fields for the NASCOM shell.  They are as
-- correct as we can make them right now.  We have had difficulty verifying the proper
-- current values for many of these fields, and a few are still set to zero because
-- we have not yet been able to find ANY value for them from JSC.  They will need to
-- be updated later.
NASCOM_Header_Sync_Byte_1                  : CONSTANT := 98;   -- 62h
NASCOM_Header_Sync_Byte_2                  : CONSTANT := 118;  -- 76h
NASCOM_Header_Sync_Byte_3                  : CONSTANT := 39;   -- 27h
Source_Code                                      : CONSTANT := 173;  -- ADh
Destination_Code                     : CONSTANT := 56;   -- 38h
Block_Sequence_Counter               : CONSTANT := 7;
Format_ID                                  : CONSTANT := 9;
User_Header_Source_Circuit_ID        : CONSTANT := 0;
User_Header_Source_Circuit_Sequence_Number  : CONSTANT := 15;
User_Header_Spare_Bit                : CONSTANT := 1;
User_Header_Block_Sequence_Number          : CONSTANT := 7;
User_Header_Message_Type                   : CONSTANT := 143; -- 8Fh
User_Header_Destination_Code         : CONSTANT := 112; -- 70h
User_Header_Spare_Bit_1                    : CONSTANT := 1;
User_Header_Spare_Bit_2                    : CONSTANT := 1;
User_Header_Full_Block_Flag                : CONSTANT := 0;
User_Header_Data_Length                    : CONSTANT := 512;
NASCOM_Trailer_F1                          : CONSTANT := 1;
NASCOM_Trailer_F2                          : CONSTANT := 0;
Command_Data_POCC_Command_Message_Number   : CONSTANT := 999;
Command_Data_Payload_Vehicle_ID            : CONSTANT := 0;
Command_Data_Command_Type                  : CONSTANT := 1;   -- 01h
Command_Data_Orbiter_Uplink_Mode           : CONSTANT := 0;
Test_Command_Command_Type                  : CONSTANT := 0;

LITE_Command_Header_SFMDM_Source_ID        : CONSTANT := 13;   -- Dh
LITE_Command_Header_SFMDM_Sync             : CONSTANT := 5;
LITE_Command_Header_SFMDM_IOM_Type         : CONSTANT := 14;   -- Eh

LITE_Command_Header_SFMDM_Command_ID       : CONSTANT := 3;
LITE_Command_Header_SFMDM_IOM              : CONSTANT := 0;
LITE_Command_Header_SFMDM_Channel_Number   : CONSTANT := 0;
LITE_Command_Header_Fill                   : CONSTANT := 0;
LITE_Command_Word_Count                    : CONSTANT := 30;
LITE_Command_Footer_Filler                 : CONSTANT := 0;

DOL_Command_Header_SFMDM_Source_ID         : CONSTANT := 13;   -- Dh
DOL_Command_Header_SFMDM_Sync              : CONSTANT := 5;
DOL_Command_Header_SFMDM_IOM_Type          : CONSTANT := 13;   -- Dh
```

```
   DOL_Command_Header_SFMDM_Command_ID         : CONSTANT := 8;
   DOL_Command_Header_SFMDM_IOM                : CONSTANT := 0;
   DOL_Command_Header_SFMDM_Channel_Number     : CONSTANT := 0;
   DOL_Command_Word_Count                      : CONSTANT := 3;
   DOL_Command_Footer_Checksum                 : CONSTANT := 244;  -- F4h

   DDCS_Command_Header_SFMDM_Source_ID         : CONSTANT := 13;   -- Dh
   DDCS_Command_Header_SFMDM_Sync              : CONSTANT := 5;
   DDCS_Command_Header_SFMDM_IOM_Type          : CONSTANT := 0;
   DDCS_Command_Header_DDCS_Command_ID         : CONSTANT := 10;  -- Ah
   DDCS_Command_Word_Count                     : CONSTANT := 8;
   DDCS_Command_Footer_Filler                  : CONSTANT := 0;


End System_Constants;
```

•

```
-----------------------------------------------------------------------------------------------------------
-- File Name:       CFGCONST.ADS (Function version)
-- Organization:    NASA - Langley Research Center (LaRC)
-- Project:         LIDAR In-Space Technology Experiment (LITE)
-----------------------------------------------------------------------------
--***************************************************************************
-- Name/Number:
--    System_Constants               (package spec)
--
-- Abstract:
--    This package spec contains several constants which are used in the
--    CFG User Interface software to define color schemes, file name
--    extensions and directories, Screen mode constants, and others.
--
-- Acronyms/Abbreviations:
--    None
--
-- Dependencies:
--    Unsigned
--    Menu_Package
--    Screen_Package
--    Keyboard_Package
--
-- Global Objects:
--    All
--
-- Exceptions:
--    None
--
-- Machine/Compiler Dependencies:
--
--***************************************************************************

with Unsigned;
with Menu_Package;
with Screen_Package;
with Keyboard_Package;


Package System_Constants_Package Is


   -- This is the main color scheme used in the menus and windows of the
   -- user interface on the Master AT.
   Function Normal_Color_Scheme RETURN Menu_Package.Color_Scheme_Type;

   -- This color scheme is used on the Master AT when a menu or window is
   -- "grayed out" after the user has made a selection.
   Function Inactive_Color_Scheme RETURN Menu_Package.Color_Scheme_Type;

   -- This color scheme is used for all CFG warning message and error
   -- message windows
   Function Warning_CScheme RETURN Menu_Package.Color_Scheme_Type;

   -- This color scheme is used for the "Emergency Go To Standby" Command Window
   Function Standby_Warning_CScheme RETURN Menu_Package.Color_Scheme_Type;

   -- These constants define the size of the screen for anybody that needs
   -- to know these things
   Function Screen_Mode RETURN Screen_Package.Text_Mode_Type;
   Function Number_of_Columns RETURN Screen_Package.Column_Type;
   Function Number_Of_Rows RETURN Screen_Package.Row_Type;
```

```
Function File_Name_Length RETURN Natural;
Function Full_File_Name_Length RETURN Natural;

-- 600 bytes in NASCOM block --
Function NASCOM_Block_Length RETURN Integer;

Function Bit_Size    RETURN Positive;
Function Nibble_Size RETURN Positive;
Function Word_Size   RETURN Positive;

Function Lookup_Table_1_Minimum_Pressure RETURN FLOAT;
Function Lookup_Table_1_Maximum_Pressure RETURN FLOAT;
Function Lookup_Table_2_Minimum_Pressure RETURN FLOAT;
Function Lookup_Table_2_Maximum_Pressure RETURN FLOAT;

-- maximum number of command words a command block can contain
Function Max_Command_Words RETURN Positive;

-- maximum number of command words in a command block transferred to
-- the IC via the PGSC

Function Max_Via_PGSC_Command_Words RETURN Positive;



-- maximum number of words in a single command block, including header
-- information words
Function Max_Block_Size  RETURN Positive;

Function S_Key_Array     RETURN Keyboard_Package.Special_Keys_Array_Type;
Function IC_S_Key_Array  RETURN Keyboard_Package.Special_Keys_Array_Type;
Function No_F1_Key_Array RETURN Keyboard_Package.Special_Keys_Array_Type;
Function F10_Key_Array   RETURN Keyboard_Package.Special_Keys_Array_Type;

Function Extra_Chars     RETURN STRING;
Function Blanks          RETURN STRING;


-- This is the maximum number of Time Specific Commands that could be in
-- a single LITE command block (if all commands were non-parametric,
-- time-specific commands, they would be 3 words each, and so at most 9 of
-- them could fit in a 28-word LITE command block).
Function Max_TS_Cmds_Per_Block RETURN Positive;
Function Max_TS_Commands       RETURN Positive;

Function Command_Menu_Width    RETURN Positive;

-- this offset is added to the hot key position read in from the data file, so that
-- all mnemonics in the menu are a few spaces over from the left edge of the menu
Function Menu_Offset           RETURN Natural;
Function Menu_Offset_String    RETURN STRING;

Function Max_Mnemonic_Length   RETURN Positive;
Function Descriptor_Length     RETURN Positive;
Function Units_String_Max      RETURN Positive;

-- Command Code constants for uplink code mode commands
Function Byte_IO_Write_Command_Code RETURN Natural;
Function Word_IO_Write_Command_Code RETURN Natural;
Function Memory_Write_Command_Code  RETURN Natural;
Function Memory_Fill_Command_Code   RETURN Natural;
```

```
-- LITE IC null commands
Function LITE_IC_Null_Cmd_Fill_Pattern_High_Byte RETURN Natural;
Function LITE_IC_Null_Cmd_Fill_Pattern_Low_Byte  RETURN Natural;

Function Reserved RETURN Unsigned.Byte;

-- row offset for next menu after drawing an inactive box around the
-- previous menu selection
Function Show_Menu_Choice_Offset RETURN Natural;

-- there are 28 Memory read strings that can be displayed at one time; this
-- area of the display is broken down into 28 rows of 45 character strings
Function Max_Memory_Reads RETURN Positive; -- up to 28 mem read displayable
Function Memory_String_Length RETURN Positive; -- string length


-- there are 45 IO read strings that can be displayed at one time; this area
-- of the display is broken down into 9 rows and 5 columns of 9 character
-- strings
Function Max_IO_Rows               RETURN Positive;  -- max IO read rows
Function Max_IO_Cols               RETURN Positive;  -- max IO read cols
Function IO_String_Length          RETURN Positive;  -- individual read string length
Function Max_IO_Reads              RETURN Positive;  -- max mem reads

Function IC_Command_Frame          RETURN Natural;
Function IC_Memory_Frame           RETURN Natural;
Function IC_Uplink_Byte            RETURN Natural;
Function Byte_IO_Read_Command_Code RETURN Natural;
Function Word_IO_Read_Command_Code RETURN Natural;
Function Memory_Read_Command_Code  RETURN Natural;

Function PDI_Sync_Pattern          RETURN Unsigned.Word;

-- min and max screen rows
Function Max_Top_Row          RETURN Screen_Package.Row_Type;
Function Max_Bottom_Row       RETURN Screen_Package.Row_Type;
Function Max_Column           RETURN Screen_Package.Column_Type;

-- number of rows in the general error window
Function Rows_In_Error_Window RETURN Screen_Package.Row_Type;

-- user input and title row offsets from top of many prompt windows
Function Input_Offset_Row     RETURN Screen_Package.Row_Type;
Function Title_Offset_Row     RETURN Screen_Package.Row_Type;

-- bottom row offset from top for many prompt windows
Function Bottom_Row_Offset    RETURN Screen_Package.Row_Type;

-- # of rows in a yes/no box
Function Yes_No_Box_Rows      RETURN Screen_Package.Row_Type;

Function Seconds_Per_Day                              RETURN Integer;
Function Seconds_Per_Hour                             RETURN Integer;

Function NASCOM_Header_Sync_1                         RETURN Integer;
Function NASCOM_Header_Sync_2                         RETURN Integer;
Function NASCOM_Header_Sync_3                         RETURN Integer;
Function Source_Code                                  RETURN Integer;
Function Destination_Code                             RETURN Integer;
Function Block_Sequence_Counter                       RETURN Integer;
Function Format_ID                                    RETURN Integer;
Function User_Header_Source_Circuit_ID                RETURN Integer;
Function User_Header_Source_Circuit_Sequence_Number   RETURN Integer;
```

```
   Function User_Header_Spare_Bit                        RETURN Integer;
   Function User_Header_Block_Sequence_Number            RETURN Integer;
   Function User_Header_Message_Type                     RETURN Integer;
   Function User_Header_Destination_Code                 RETURN Integer;
   Function User_Header_Spare_Bit_1                      RETURN Integer;
   Function User_Header_Spare_Bit_2                      RETURN Integer;
   Function User_Header_Full_Block_Flag                  RETURN Integer;
   Function User_Header_Data_Length                      RETURN Integer;
   Function NASCOM_Trailer_F1                            RETURN Integer;
   Function NASCOM_Trailer_F2                            RETURN Integer;
   Function Command_Data_POCC_Command_Message_Number     RETURN Integer;
   Function Command_Data_Payload_Vehicle_ID              RETURN Integer;
   Function Command_Data_Command_Type                    RETURN Integer;
   Function Command_Data_Orbiter_Uplink_Mode             RETURN Integer;
   Function Test_Command_Command_Type                    RETURN Integer;

   Function LITE_Command_Header_SFMDM_Source_ID          RETURN Integer;
   Function LITE_Command_Header_SFMDM_Sync               RETURN Integer;
   Function LITE_Command_Header_SFMDM_IOM_Type           RETURN Integer;
   Function LITE_Command_Header_SFMDM_Command_ID         RETURN Integer;
   Function LITE_Command_Header_SFMDM_IOM                RETURN Integer;
   Function LITE_Command_Header_SFMDM_Channel_Number     RETURN Integer;
   Function LITE_Command_Header_Fill                     RETURN Integer;
   Function LITE_Command_Word_Count                      RETURN Integer;
   Function LITE_Command_Footer_Filler                   RETURN Integer;

   Function DOL_Command_Header_SFMDM_Source_ID           RETURN Integer;
   Function DOL_Command_Header_SFMDM_Sync                RETURN Integer;
   Function DOL_Command_Header_SFMDM_IOM_Type            RETURN Integer;
   Function DOL_Command_Header_SFMDM_Command_ID          RETURN Integer;
   Function DOL_Command_Header_SFMDM_IOM                 RETURN Integer;
   Function DOL_Command_Header_SFMDM_Channel_Number      RETURN Integer;
   Function DOL_Command_Word_Count                       RETURN Integer;
   Function DOL_Command_Footer_Checksum                  RETURN Integer;

   Function DDCS_Command_Header_SFMDM_Source_ID          RETURN Integer;
   Function DDCS_Command_Header_SFMDM_Sync               RETURN Integer;
   Function DDCS_Command_Header_SFMDM_IOM_Type           RETURN Integer;
   Function DDCS_Command_Header_DDCS_Command_ID          RETURN Integer;
   Function DDCS_Command_Word_Count                      RETURN Integer;
   Function DDCS_Command_Footer_Filler                   RETURN Integer;

End System_Constants_Package;
```

•

```
-------------------------------------------------------------------------------
-- File Name:       CFGCONST.ADB
-- Organization:    NASA - Langley Research Center (LaRC)
-- Project:         LIDAR In-Space Technology Experiment (LITE)
-------------------------------------------------------------------------------
--*****************************************************************************
-- Name/Number:
--    System_Constants_Package               (package body)
--
-- Abstract:
--    This package spec contains several constants which are used in the
--    CFG User Interface software to define color schemes, file name
--    extensions and directories, Screen mode constants, and others.  It
--    has no body.
--
-- Acronyms/Abbreviations:
--    None
--
-- Dependencies:
--    Unsigned
--    Menu_Package
--    Screen_Package
--    Keyboard_Package
--
-- Global Objects:
--    All
--
-- Exceptions:
--    None
--
-- Machine/Compiler Dependencies:
--
--*****************************************************************************

with Unsigned;
with Menu_Package;
with Screen_Package;
with Keyboard_Package;


Package Body System_Constants_Package Is


   -- This is the main color scheme used in the menus and windows of the
   -- user interface on the Master AT.
   Normal_Color_Scheme_Val : CONSTANT Menu_Package.Color_Scheme_Type :=
      (Screen_Package.Blue_Background,         -- Background_Color
       Screen_Package.Bright_White_Foreground, -- Foreround_Color
       Screen_Package.Yellow_Foreground,       -- Title_Color
       Screen_Package.Yellow_Foreground,       -- Border_Color
       Screen_Package.Light_Red_Foreground);   -- Highlight_Color

   -- This color scheme is used on the Master AT when a menu or window is
   -- "grayed out" after the user has made a selection.
   Inactive_Color_Scheme_Val : CONSTANT Menu_Package.Color_Scheme_Type :=
      (Screen_Package.Blue_Background,         -- Background_Color
       Screen_Package.White_Foreground,        -- Foreround_Color
       Screen_Package.Yellow_Foreground,       -- Title_Color
       Screen_Package.White_Foreground,        -- Border_Color
       Screen_Package.Black_Foreground);       -- Highlight_Color

   -- This color scheme is used for all CFG warning message and error
   -- message windows
```

```
Warning_CScheme_Val : CONSTANT Menu_Package.Color_Scheme_Type :=
   (Screen_Package.Red_Background,          -- Background_Color
    Screen_Package.Bright_White_Foreground, -- Foreround_Color
    Screen_Package.Yellow_Foreground,       -- Title_Color
    Screen_Package.Yellow_Foreground,       -- Border_Color
    Screen_Package.Yellow_Foreground);      -- Highlight_Color

-- This color scheme is used for the "Emergency Go To Standby" Command Window
Standby_Warning_CScheme_Val : CONSTANT Menu_Package.Color_Scheme_Type :=
   (Screen_Package.Magenta_Background,      -- Background_Color
    Screen_Package.Light_Green_Foreground,  -- Foreround_Color
    Screen_Package.Bright_White_Foreground, -- Title_Color
    Screen_Package.Yellow_Foreground,       -- Border_Color
    Screen_Package.Yellow_Foreground);      -- Highlight_Color

-- These constants define the size of the screen for anybody that needs
-- to know these things
Screen_Mode_Val       : CONSTANT Screen_Package.Text_Mode_Type :=
                        Screen_Package.Color_80x50;
Number_of_Columns_Val : CONSTANT Screen_Package.Column_Type    := 80;
Number_Of_Rows_Val    : CONSTANT Screen_Package.Row_Type       := 50;

File_Name_Length_Val      : CONSTANT Natural := 8;
Full_File_Name_Length_Val : CONSTANT Natural := 12; --this includes the file extension

-- 600 bytes in NASCOM block --
NASCOM_Block_Length_Val             : CONSTANT Integer := 600;

Bit_Size_Val                        : CONSTANT Positive := 1;
Nibble_Size_Val                     : CONSTANT Positive := 4;
Word_Size_Val                       : CONSTANT Positive := 16;

Lookup_Table_1_Minimum_Pressure_Val : CONSTANT FLOAT := 15.0;
Lookup_Table_1_Maximum_Pressure_Val : CONSTANT FLOAT := 65.0;
Lookup_Table_2_Minimum_Pressure_Val : CONSTANT FLOAT :=  5.0;
Lookup_Table_2_Maximum_Pressure_Val : CONSTANT FLOAT := 65.0;

-- maximum number of command words a command block can contain
Max_Command_Words_Val               : CONSTANT Positive := 28;

-- maximum number of command words in a command block transferred to
-- the IC via the PGSC
Max_Via_PGSC_Command_Words_Val      : CONSTANT Positive := 22;

-- maximum number of words in a single command block, including header
-- information words
Max_Block_Size_Val                  : CONSTANT Positive := 30;

S_Key_Array_Val     : CONSTANT Keyboard_Package.Special_Keys_Array_Type (1..3) :=
                      (Keyboard_Package.F1_Key, Keyboard_Package.F2_Key,
                       Keyboard_Package.F10_Key);
IC_S_Key_Array_Val  : CONSTANT Keyboard_Package.Special_Keys_Array_Type (1..4) :=
                      (Keyboard_Package.F1_Key, Keyboard_Package.F2_Key,
                       Keyboard_Package.F3_Key, Keyboard_Package.F10_Key);
No_F1_Key_Array_Val : CONSTANT Keyboard_Package.Special_Keys_Array_Type (1..2) :=
                      (Keyboard_Package.F2_Key, Keyboard_Package.F10_Key);
F10_Key_Array_Val   : CONSTANT Keyboard_Package.Special_Keys_Array_Type (1..1) :=
                      (OTHERS => Keyboard_Package.F10_Key);

Extra_Chars_Val     : CONSTANT STRING := " .,<>?/\';:[]{}-_=+|`~!@#$%^&*()";
Blanks_Val          : CONSTANT STRING (1 .. 80) := (OTHERS => ' ');
```

```
-- This is the maximum number of Time Specific Commands that could be in
-- a single LITE command block (if all commands were non-parametric,
-- time-specific commands, they would be 3 words each, and so at most 9 of
-- them could fit in a 28-word LITE command block).
Max_TS_Cmds_Per_Block_Val : CONSTANT Positive := 9;
Max_TS_Commands_Val       : CONSTANT Positive := 32;

Command_Menu_Width_Val    : CONSTANT Positive := 24;

-- this offset is added to the hot key position read in from the data file, so that
-- all mnemonics in the menu are a few spaces over from the left edge of the menu
Menu_Offset_Val           : CONSTANT Natural := 3;
Menu_Offset_String_Val    : CONSTANT STRING (1..3) := (OTHERS => ' ');

Max_Mnemonic_Length_Val   : CONSTANT Positive := 15;
Descriptor_Length_Val     : CONSTANT Positive := 40;
Units_String_Max_Val      : CONSTANT Positive := 5;

-- Command Code constants for uplink code mode commands
Byte_IO_Write_Command_Code_VAL : CONSTANT Natural := 16#50#;
Word_IO_Write_Command_Code_Val : CONSTANT Natural := 16#51#;
Memory_Write_Command_Code_Val  : CONSTANT Natural := 16#54#;
Memory_Fill_Command_Code_Val   : CONSTANT Natural := 16#56#;

-- LITE IC null commands
LITE_IC_Null_Cmd_Fill_Pattern_High_Byte_Val : CONSTANT Natural := 16#04#;
LITE_IC_Null_Cmd_Fill_Pattern_Low_Byte_Val  : CONSTANT Natural := 16#FF#;

Reserved_Val                   : CONSTANT Unsigned.Byte := 0;

-- row offset for next menu after drawing an inactive box around the
-- previous menu selection
Show_Menu_Choice_Offset_Val    : CONSTANT Natural := 5;

-- there are 28 Memory read strings that can be displayed at one time; this
-- area of the display is broken down into 28 rows of 45 character strings
Max_Memory_Reads_Val     : CONSTANT Positive := 28; -- up to 28 mem read displayable
Memory_String_Length_Val : CONSTANT Positive := 45; -- string length

-- there are 45 IO read strings that can be displayed at one time; this area
-- of the display is broken down into 9 rows and 5 columns of 9 character
-- strings
Max_IO_Rows_Val     : CONSTANT POSITIVE := 9;  -- max IO read rows
Max_IO_Cols_Val     : CONSTANT POSITIVE := 5;  -- max IO read cols
IO_String_Length_Val: CONSTANT POSITIVE := 9;  -- individual read string length
Max_IO_Reads_Val    : CONSTANT POSITIVE := Max_IO_Rows * Max_IO_Cols; -- max mem reads

IC_Command_Frame_Val           : CONSTANT Natural := 8;
IC_Memory_Frame_Val            : CONSTANT Natural := 10;
IC_Uplink_Byte_Val             : CONSTANT Natural := 53;
Byte_IO_Read_Command_Code_Val  : CONSTANT Natural := 16#52#;
Word_IO_Read_Command_Code_Val  : CONSTANT Natural := 16#53#;
Memory_Read_Command_Code_Val   : CONSTANT Natural := 16#55#;

PDI_Sync_Pattern_Val           : CONSTANT Unsigned.Word := 16#BEEF#;

-- min and max screen rows
Max_Top_Row_Val          : CONSTANT Screen_Package.Row_Type    := 1;
Max_Bottom_Row_Val       : CONSTANT Screen_Package.Row_Type    := 49;
Max_Column_Val           : CONSTANT Screen_Package.Column_Type := 79;

-- number of rows in the general error window
Rows_In_Error_Window_Val : CONSTANT Screen_Package.Row_Type    := 5;
```

```
-- user input and title row offsets from top of many prompt windows
Input_Offset_Row_Val     : CONSTANT Screen_Package.Row_Type    := 2;
Title_Offset_Row_Val     : CONSTANT Screen_Package.Row_Type    := 2;

-- bottom row offset from top for many prompt windows
Bottom_Row_Offset_Val    : CONSTANT Screen_Package.Row_Type    := 4;

-- # of rows in a yes/no box
Yes_No_Box_Rows_Val      : CONSTANT Screen_Package.Row_Type    := 6;

Seconds_Per_Day_Val      : CONSTANT Integer := 86400;
Seconds_Per_Hour_Val     : CONSTANT Integer := 3600;

NASCOM_Header_Sync_Byte_1_Val                  : CONSTANT Integer := 98;  -- 62h
NASCOM_Header_Sync_Byte_2_Val                  : CONSTANT Integer := 118; -- 76h
NASCOM_Header_Sync_Byte_3_Val                  : CONSTANT Integer := 39;  -- 27h
Source_Code_Val                                : CONSTANT Integer := 173; -- ADh
Destination_Code_Val                           : CONSTANT Integer := 56;  -- 38h
Block_Sequence_Counter_Val                     : CONSTANT Integer := 7;
Format_ID_Val                                  : CONSTANT Integer := 9;
User_Header_Source_Circuit_ID_Val              : CONSTANT Integer := 255;
User_Header_Source_Circuit_Sequence_Number_Val : CONSTANT Integer := 15;
User_Header_Spare_Bit_Val                      : CONSTANT Integer := 1;
User_Header_Block_Sequence_Number_Val          : CONSTANT Integer := 7;
User_Header_Message_Type_Val                   : CONSTANT Integer := 143; -- 8Fh
User_Header_Destination_Code_Val               : CONSTANT Integer := 112; -- 70h
User_Header_Spare_Bit_1_Val                    : CONSTANT Integer := 1;
User_Header_Spare_Bit_2_Val                    : CONSTANT Integer := 1;
User_Header_Full_Block_Flag_Val                : CONSTANT Integer := 0;
User_Header_Data_Length_Val                    : CONSTANT Integer := 512;
NASCOM_Trailer_F1_Val                          : CONSTANT Integer := 1;
NASCOM_Trailer_F2_Val                          : CONSTANT Integer := 0;
Command_Data_POCC_Command_Message_Number_Val   : CONSTANT Integer := 999;
Command_Data_Payload_Vehicle_ID_Val            : CONSTANT Integer := 55;  -- 37h
Command_Data_Command_Type_Val                  : CONSTANT Integer := 1;   -- 01h
Command_Data_Orbiter_Uplink_Mode_Val           : CONSTANT Integer := 4;   -- 04h
Test_Command_Command_Type_Val                  : CONSTANT Integer := 0;   --  0h

LITE_Command_Header_SFMDM_Source_ID_Val        : CONSTANT Integer := 13;   -- Dh
LITE_Command_Header_SFMDM_Sync_Val             : CONSTANT Integer := 5;
LITE_Command_Header_SFMDM_IOM_Type_Val         : CONSTANT Integer := 14;   -- Eh
LITE_Command_Header_SFMDM_Command_ID_Val       : CONSTANT Integer := 3;
LITE_Command_Header_SFMDM_IOM_Val              : CONSTANT Integer := 0;
LITE_Command_Header_SFMDM_Channel_Number_Val   : CONSTANT Integer := 0;
LITE_Command_Header_Fill_Val                   : CONSTANT Integer := 0;
LITE_Command_Word_Count_Val                    : CONSTANT Integer := 30;
LITE_Command_Footer_Filler_Val                 : CONSTANT Integer := 0;

DOL_Command_Header_SFMDM_Source_ID_Val         : CONSTANT Integer := 13;   -- Dh
DOL_Command_Header_SFMDM_Sync_Val              : CONSTANT Integer := 5;
DOL_Command_Header_SFMDM_IOM_Type_Val          : CONSTANT Integer := 13;   -- Dh
DOL_Command_Header_SFMDM_Command_ID_Val        : CONSTANT Integer := 8;
DOL_Command_Header_SFMDM_IOM_Val               : CONSTANT Integer := 0;
DOL_Command_Header_SFMDM_Channel_Number_Val    : CONSTANT Integer := 0;
DOL_Command_Word_Count_Val                     : CONSTANT Integer := 3;
DOL_Command_Footer_Checksum_Val                : CONSTANT Integer := 244;  -- F4h

DDCS_Command_Header_SFMDM_Source_ID_Val        : CONSTANT Integer := 13;   -- Dh
DDCS_Command_Header_SFMDM_Sync_Val             : CONSTANT Integer := 5;
DDCS_Command_Header_SFMDM_IOM_Type_Val         : CONSTANT Integer := 0;
DDCS_Command_Header_DDCS_Command_ID_Val        : CONSTANT Integer := 10;  -- Ah
DDCS_Command_Word_Count_Val                    : CONSTANT Integer := 8;
DDCS_Command_Footer_Filler_Val                 : CONSTANT Integer := 0;
```

```
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
-- "constant" functions
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
-- First, some user interface constants, starting with...
-- color schemes
--------------------------------------------------------------------------------

   -- This is the main color scheme used in the menus and windows of the
   -- user interface on the Master AT.
   Function Normal_Color_Scheme RETURN Menu_Package.Color_Scheme_Type IS
   BEGIN
      RETURN Normal_Color_Scheme_Val;
   END Normal_Color_Scheme;

   -- This color scheme is used on the Master AT when a menu or window is
   -- "grayed out" after the user has made a selection.
   Function Inactive_Color_Scheme RETURN Menu_Package.Color_Scheme_Type IS
   BEGIN
      RETURN Inactive_Color_Scheme_Val;
   END Inactive_Color_Scheme;

   -- This color scheme is used for all CFG warning message and error
   -- message windows
   Function Warning_CScheme RETURN Menu_Package.Color_Scheme_Type IS
   BEGIN
      RETURN Warning_CScheme_Val;

   END Warning_CScheme;

   -- This color scheme is used for the "Emergency Go To Standby" Command Window
   Function Standby_Warning_CScheme RETURN Menu_Package.Color_Scheme_Type IS
   BEGIN
      RETURN Standby_Warning_CScheme_Val;
   END Standby_Warning_CScheme;


--------------------------------------------------------------------------------
-- ... next, some screen locations and important columns and rows:

--------------------------------------------------------------------------------



   -- These constants define the size of the screen for anybody that needs
   -- to know these things

   Function Screen_Mode RETURN Screen_Package.Text_Mode_Type IS
   BEGIN
      RETURN Screen_Mode_Val;
   END Screen_Mode;

   Function Number_of_Columns RETURN Screen_Package.Column_Type IS
   BEGIN
      RETURN Number_of_Columns_Val;
   END Number_of_Columns;

   Function Number_Of_Rows RETURN Screen_Package.Row_Type IS
   BEGIN
```

```
   RETURN Number_Of_Rows_Val;
END Number_Of_Rows;


-- min and max screen rows
Function Max_Top_Row RETURN Screen_Package.Row_Type IS
BEGIN
   RETURN Max_Top_Row_Val;
END Max_Top_Row;

Function Max_Bottom_Row RETURN Screen_Package.Row_Type IS
BEGIN
   RETURN Max_Bottom_Row_Val;
END Max_Bottom_Row;

Function Max_Column RETURN Screen_Package.Column_Type IS
BEGIN
   RETURN Max_Column_Val;
END Max_Column;

-- number of rows in the general error window
Function Rows_In_Error_Window RETURN Screen_Package.Row_Type IS
BEGIN
   RETURN Rows_In_Error_Window_Val;
END Rows_In_Error_Window;

-- user input and title row offsets from top of many prompt windows
Function Input_Offset_Row RETURN Screen_Package.Row_Type IS
BEGIN
   RETURN Input_Offset_Row_Val;
END Input_Offset_Row;

Function Title_Offset_Row RETURN Screen_Package.Row_Type IS
BEGIN
   RETURN Title_Offset_Row_Val;
END Title_Offset_Row;

-- bottom row offset from top for many prompt windows
Function Bottom_Row_Offset RETURN Screen_Package.Row_Type IS
BEGIN
   RETURN Bottom_Row_Offset_Val;
END Bottom_Row_Offset;

-- # of rows in a yes/no box
Function Yes_No_Box_Rows RETURN Screen_Package.Row_Type IS
BEGIN
   RETURN Yes_No_Box_Rows_Val;
END Yes_No_Box_Rows;

-- row offset for next menu after drawing an inactive box around the
-- previous menu selection
Function Show_Menu_Choice_Offset RETURN Natural IS
BEGIN
   RETURN Show_Menu_Choice_Offset_Val;
END Show_Menu_Choice_Offset;

Function Command_Menu_Width RETURN Positive IS
BEGIN
   RETURN Command_Menu_Width_Val;
END Command_Menu_Width;
```

```
-------------------------------------------------------------------------------
-- ... next, some keyboard arrays of special keys to be used by the user:
-------------------------------------------------------------------------------

   Function S_Key_Array RETURN Keyboard_Package.Special_Keys_Array_Type IS
   BEGIN
      RETURN S_Key_Array_Val;
   END S_Key_Array;

   Function IC_S_Key_Array RETURN Keyboard_Package.Special_Keys_Array_Type IS
   BEGIN
      RETURN IC_S_Key_Array_Val;
   END IC_S_Key_Array;

   Function No_F1_Key_Array RETURN Keyboard_Package.Special_Keys_Array_Type IS
   BEGIN
      RETURN No_F1_Key_Array_Val;
   END No_F1_Key_Array;

   Function F10_Key_Array RETURN Keyboard_Package.Special_Keys_Array_Type IS
   BEGIN
      RETURN F10_Key_Array_Val;
   End F10_Key_Array;

-------------------------------------------------------------------------------
-- ... and these are some general-purpose constants used in the user interface:
-------------------------------------------------------------------------------

   Function Extra_Chars RETURN STRING IS
   BEGIN
      RETURN Extra_Chars_Val;
   END Extra_Chars;

   Function Blanks RETURN STRING IS
   BEGIN
      RETURN Blanks_Val;
   END Blanks;

   -- this offset is added to the hot key position read in from the data file, so that
   -- all mnemonics in the menu are a few spaces over from the left edge of the menu
   Function Menu_Offset RETURN Natural IS
   BEGIN
      RETURN Menu_Offset_Val;
   END Menu_Offset;

   Function Menu_Offset_String RETURN STRING IS
   BEGIN
      RETURN Menu_Offset_String_Val;
   END Menu_Offset_String;

   Function Max_Mnemonic_Length RETURN Positive IS
   BEGIN
      RETURN Max_Mnemonic_Length_Val;
   END Max_Mnemonic_Length;

   Function Descriptor_Length RETURN Positive IS
   BEGIN
      RETURN Descriptor_Length_Val;
   END Descriptor_Length;

   Function Units_String_Max RETURN Positive IS
   BEGIN
      RETURN Units_String_Max_Val;
```

```
   END Units_String_Max;

   -- maximum number of command words a command block can contain
   Function Max_Command_Words RETURN Positive IS
   BEGIN
      RETURN Max_Command_Words_Val;
   END Max_Command_Words;

   -- maximum number of command words in a command block transferred to
   -- the IC via the PGSC
   Function Max_Via_PGSC_Command_Words RETURN Positive IS
   BEGIN
      RETURN Max_Via_PGSC_Command_Words_Val;
   END Max_Via_PGSC_Command_Words;

   -- maximum number of words in a single command block, including header
   -- information words
   Function Max_Block_Size RETURN Positive IS
   BEGIN
      RETURN Max_Block_Size_Val;
   END Max_Block_Size;

   -- This is the maximum number of Time Specific Commands that could be in
   -- a single LITE command block (if all commands were non-parametric,
   -- time-specific commands, they would be 3 words each, and so at most 9 of
   -- them could fit in a 28-word LITE command block).
   Function Max_TS_Cmds_Per_Block RETURN Positive IS
   BEGIN
      RETURN Max_TS_Cmds_Per_Block_Val;
   END Max_TS_Cmds_Per_Block;

   Function Max_TS_Commands RETURN Positive IS
   BEGIN
      RETURN Max_TS_Commands_Val;
   END Max_TS_Commands;


-------------------------------------------------------------------------------
-- Now, a bunch of constants related to Uplink Code Mode:
-------------------------------------------------------------------------------

   -- Command Code constants for uplink code mode commands
   Function Byte_IO_Write_Command_Code RETURN Natural IS
   BEGIN
      RETURN Byte_IO_Write_Command_Code_Val;
   END Byte_IO_Write_Command_Code;

   Function Word_IO_Write_Command_Code RETURN Natural IS
   BEGIN
      RETURN Word_IO_Write_Command_Code_Val;
   END Word_IO_Write_Command_Code;

   Function Byte_IO_Read_Command_Code RETURN Natural IS
   BEGIN
      RETURN Byte_IO_Read_Command_Code_Val;
   END Byte_IO_Read_Command_Code;

   Function Word_IO_Read_Command_Code RETURN Natural IS
   BEGIN
      RETURN Word_IO_Read_Command_Code_Val;
   END Word_IO_Read_Command_Code;

   Function Memory_Write_Command_Code RETURN Natural IS
```

```
BEGIN
   RETURN Memory_Write_Command_Code_Val;
END Memory_Write_Command_Code;

Function Memory_Read_Command_Code RETURN Natural IS
BEGIN
   RETURN Memory_Read_Command_Code_Val;
END Memory_Read_Command_Code;

Function Memory_Fill_Command_Code RETURN Natural IS
BEGIN
   RETURN Memory_Fill_Command_Code_Val;
END Memory_Fill_Command_Code;

-- there are 28 Memory read strings that can be displayed at one time; this
-- area of the display is broken down into 28 rows of 45 character strings
Function Max_Memory_Reads RETURN Positive IS
BEGIN
   RETURN Max_Memory_Reads_Val;
END Max_Memory_Reads;

Function Memory_String_Length RETURN Positive IS
BEGIN
   RETURN Memory_String_Length_Val;
END Memory_String_Length;

-- there are 45 IO read strings that can be displayed at one time; this area
-- of the display is broken down into 9 rows and 5 columns of 9 character
-- strings

Function Max_IO_Rows RETURN Positive IS
BEGIN
   RETURN Max_IO_Rows_Val;
END Max_IO_Rows;

Function Max_IO_Cols RETURN Positive IS
BEGIN
   RETURN Max_IO_Cols_Val;
END Max_IO_Cols;

Function IO_String_Length RETURN Positive IS
BEGIN
   RETURN IO_String_Length_Val;
END IO_String_Length;

Function Max_IO_Reads RETURN Positive IS
BEGIN
   RETURN Max_IO_Reads_Val;
END Max_IO_Reads;

Function IC_Command_Frame RETURN Natural IS
BEGIN
   RETURN IC_Command_Frame_Val;
END IC_Command_Frame;

Function IC_Memory_Frame RETURN Natural IS
BEGIN
   RETURN IC_Memory_Frame_Val;
END IC_Memory_Frame;

Function IC_Uplink_Byte RETURN Natural IS
BEGIN
   RETURN IC_Uplink_Byte_Val;
```

```
   END IC_Uplink_Byte;


--------------------------------------------------------------------------------
-- these are used to compute the parameter for lookup table commands
--------------------------------------------------------------------------------

   Function Lookup_Table_1_Minimum_Pressure RETURN FLOAT IS
   BEGIN
      RETURN Lookup_Table_1_Minimum_Pressure_Val;
   END Lookup_Table_1_Minimum_Pressure;

   Function Lookup_Table_1_Maximum_Pressure RETURN FLOAT IS
   BEGIN
      RETURN Lookup_Table_1_Maximum_Pressure_Val;
   END Lookup_Table_1_Maximum_Pressure;

   Function Lookup_Table_2_Minimum_Pressure RETURN FLOAT IS
   BEGIN
      RETURN Lookup_Table_2_Minimum_Pressure_Val;
   END Lookup_Table_2_Minimum_Pressure;

   Function Lookup_Table_2_Maximum_Pressure RETURN FLOAT IS
   BEGIN
      RETURN Lookup_Table_2_Maximum_Pressure_Val;
   END Lookup_Table_2_Maximum_Pressure;

--------------------------------------------------------------------------------
-- some useful general purpose constants:
--------------------------------------------------------------------------------

   Function Bit_Size RETURN Positive IS
   BEGIN
      RETURN Bit_Size_Val;
   END Bit_Size;

   Function Nibble_Size RETURN Positive IS
   BEGIN
      RETURN Nibble_Size_Val;
   END Nibble_Size;

   Function Word_Size RETURN Positive IS
   BEGIN
      RETURN Word_Size_Val;
   END Word_Size;

   Function File_Name_Length RETURN Natural IS
   BEGIN
      RETURN File_Name_Length_Val;
   END File_Name_Length;

   Function Full_File_Name_Length RETURN Natural IS
   BEGIN
      RETURN Full_File_Name_Length_Val;
   END Full_File_Name_Length;

   Function Seconds_Per_Day RETURN Integer IS
   BEGIN
      RETURN Seconds_Per_Day_Val;
   END Seconds_Per_Day;

   Function Seconds_Per_Hour RETURN Integer IS
   BEGIN
```

```
      RETURN Seconds_Per_Hour_Val;
   END Seconds_Per_Hour;

-------------------------------------------------------------------------------
-- constants required to build the NASCOM block:
-------------------------------------------------------------------------------

   Function PDI_Sync_Pattern RETURN Unsigned.Word IS
   BEGIN
      RETURN PDI_Sync_Pattern_Val;
   END PDI_Sync_Pattern;

   -- 600 bytes in NASCOM block --
   Function NASCOM_Block_Length RETURN Integer IS
   BEGIN
      RETURN NASCOM_Block_Length_Val;
   END NASCOM_Block_Length;

   Function NASCOM_Header_Sync_1 RETURN Integer IS
   BEGIN
      RETURN NASCOM_Header_Sync_Byte_1_Val;
   END NASCOM_Header_Sync_1;

   Function NASCOM_Header_Sync_2 RETURN Integer IS
   BEGIN
      RETURN NASCOM_Header_Sync_Byte_2_Val;
   END NASCOM_Header_Sync_2;

   Function NASCOM_Header_Sync_3 RETURN Integer IS
   BEGIN
      RETURN NASCOM_Header_Sync_Byte_3_Val;
   END NASCOM_Header_Sync_3;

   Function Source_Code RETURN Integer IS
   BEGIN
      RETURN Source_Code_Val;
   END Source_Code;

   Function Destination_Code RETURN Integer IS
   BEGIN
      RETURN Destination_Code_Val;
   END Destination_Code;

   Function Block_Sequence_Counter RETURN Integer IS
   BEGIN
      RETURN Block_Sequence_Counter_Val;
   END Block_Sequence_Counter;

   Function Format_ID RETURN Integer IS
   BEGIN
      RETURN Format_ID_Val;
   END Format_ID;

   Function User_Header_Source_Circuit_ID RETURN Integer IS
   BEGIN
      RETURN User_Header_Source_Circuit_ID_Val;
   END User_Header_Source_Circuit_ID;

   Function User_Header_Source_Circuit_Sequence_Number RETURN Integer IS
   BEGIN
      RETURN User_Header_Source_Circuit_Sequence_Number_Val;
   END User_Header_Source_Circuit_Sequence_Number;
```

```
Function User_Header_Spare_Bit RETURN Integer IS
BEGIN
   RETURN User_Header_Spare_Bit_Val;
END User_Header_Spare_Bit;

Function User_Header_Block_Sequence_Number RETURN Integer IS
BEGIN
   RETURN User_Header_Block_Sequence_Number_Val;
END User_Header_Block_Sequence_Number;

Function User_Header_Message_Type RETURN Integer IS
BEGIN
   RETURN User_Header_Message_Type_Val;
END User_Header_Message_Type;

Function User_Header_Destination_Code RETURN Integer IS
BEGIN
   RETURN User_Header_Destination_Code_Val;
END User_Header_Destination_Code;

Function User_Header_Spare_Bit_1 RETURN Integer IS
BEGIN
   RETURN User_Header_Spare_Bit_1_Val;
END User_Header_Spare_Bit_1;

Function User_Header_Spare_Bit_2 RETURN Integer IS
BEGIN
   RETURN User_Header_Spare_Bit_2_Val;
END User_Header_Spare_Bit_2;

Function User_Header_Full_Block_Flag RETURN Integer IS
BEGIN
   RETURN User_Header_Full_Block_Flag_Val;
END User_Header_Full_Block_Flag;

Function User_Header_Data_Length RETURN Integer IS
BEGIN
   RETURN User_Header_Data_Length_Val;
END User_Header_Data_Length;

Function NASCOM_Trailer_F1 RETURN Integer IS
BEGIN
   RETURN NASCOM_Trailer_F1_Val;
END NASCOM_Trailer_F1;

Function NASCOM_Trailer_F2 RETURN Integer IS
BEGIN
   RETURN NASCOM_Trailer_F2_Val;
END NASCOM_Trailer_F2;

Function Command_Data_POCC_Command_Message_Number RETURN Integer IS
BEGIN
   RETURN Command_Data_POCC_Command_Message_Number_Val;
END Command_Data_POCC_Command_Message_Number;

Function Command_Data_Payload_Vehicle_ID RETURN Integer IS
BEGIN
   RETURN Command_Data_Payload_Vehicle_ID_Val;
END Command_Data_Payload_Vehicle_ID;

Function Command_Data_Command_Type RETURN Integer IS
BEGIN
   RETURN Command_Data_Command_Type_Val;
```

```
   END Command_Data_Command_Type;

   Function Command_Data_Orbiter_Uplink_Mode RETURN Integer IS
   BEGIN
      RETURN Command_Data_Orbiter_Uplink_Mode_Val;
   END Command_Data_Orbiter_Uplink_Mode;

   Function Test_Command_Command_Type RETURN Integer IS
   BEGIN
      RETURN Test_Command_Command_Type_Val;
   END Test_Command_Command_Type;

--------------------------------------------------------------------------------
-- constants required to fill the LITE commadn block header and footer:
--------------------------------------------------------------------------------

   Function LITE_Command_Header_SFMDM_Source_ID RETURN Integer IS
   BEGIN
      RETURN LITE_Command_Header_SFMDM_Source_ID_Val;
   END LITE_Command_Header_SFMDM_Source_ID;

   Function LITE_Command_Header_SFMDM_Sync RETURN Integer IS
   BEGIN
      RETURN LITE_Command_Header_SFMDM_Sync_Val;
   END LITE_Command_Header_SFMDM_Sync;

   Function LITE_Command_Header_SFMDM_IOM_Type RETURN Integer IS
   BEGIN
      RETURN LITE_Command_Header_SFMDM_IOM_Type_Val;
   END LITE_Command_Header_SFMDM_IOM_Type;

   Function LITE_Command_Header_SFMDM_Command_ID RETURN Integer IS
   BEGIN
      RETURN LITE_Command_Header_SFMDM_Command_ID_Val;
   END LITE_Command_Header_SFMDM_Command_ID;

   Function LITE_Command_Header_SFMDM_IOM RETURN Integer IS
   BEGIN
      RETURN LITE_Command_Header_SFMDM_IOM_Val;
   END LITE_Command_Header_SFMDM_IOM;

   Function LITE_Command_Header_SFMDM_Channel_Number RETURN Integer IS
   BEGIN
      RETURN LITE_Command_Header_SFMDM_Channel_Number_Val;
   END LITE_Command_Header_SFMDM_Channel_Number;

   Function LITE_Command_Header_Fill RETURN Integer IS
   BEGIN
      RETURN LITE_Command_Header_Fill_Val;
   END LITE_Command_Header_Fill;

   Function LITE_Command_Word_Count RETURN Integer IS
   BEGIN
      RETURN LITE_Command_Word_Count_Val;
   END LITE_Command_Word_Count;

   Function LITE_Command_Footer_Filler RETURN Integer IS
   BEGIN
      RETURN LITE_Command_Footer_Filler_Val;
   END LITE_Command_Footer_Filler;

   Function Reserved RETURN Unsigned.Byte IS
   BEGIN
```

```
      RETURN Reserved_Val;
   END Reserved;

   -- LITE IC null commands
   Function LITE_IC_Null_Cmd_Fill_Pattern_High_Byte RETURN Natural IS
   BEGIN
      RETURN LITE_IC_Null_Cmd_Fill_Pattern_High_Byte_Val;
   END LITE_IC_Null_Cmd_Fill_Pattern_High_Byte;

   Function LITE_IC_Null_Cmd_Fill_Pattern_Low_Byte RETURN Natural IS
   BEGIN
      RETURN LITE_IC_Null_Cmd_Fill_Pattern_Low_Byte_Val;
   END LITE_IC_Null_Cmd_Fill_Pattern_Low_Byte;



--------------------------------------------------------------------------------
-- constants used to configure and send DOL commands in a NASCOM block
--------------------------------------------------------------------------------

   Function DOL_Command_Header_SFMDM_Source_ID RETURN Integer IS
   BEGIN
      RETURN DOL_Command_Header_SFMDM_Source_ID_Val;
   END DOL_Command_Header_SFMDM_Source_ID;

   Function DOL_Command_Header_SFMDM_Sync RETURN Integer IS
   BEGIN
      RETURN DOL_Command_Header_SFMDM_Sync_Val;
   END DOL_Command_Header_SFMDM_Sync;

   Function DOL_Command_Header_SFMDM_IOM_Type RETURN Integer IS
   BEGIN
      RETURN DOL_Command_Header_SFMDM_IOM_Type_Val;
   END DOL_Command_Header_SFMDM_IOM_Type;

   Function DOL_Command_Header_SFMDM_Command_ID RETURN Integer IS
   BEGIN
      RETURN DOL_Command_Header_SFMDM_Command_ID_Val;
   END DOL_Command_Header_SFMDM_Command_ID;

   Function DOL_Command_Header_SFMDM_IOM RETURN Integer IS
   BEGIN
      RETURN DOL_Command_Header_SFMDM_IOM_Val;
   END DOL_Command_Header_SFMDM_IOM;

   Function DOL_Command_Header_SFMDM_Channel_Number RETURN Integer IS
   BEGIN
      RETURN DOL_Command_Header_SFMDM_Channel_Number_Val;
   END DOL_Command_Header_SFMDM_Channel_Number;

   Function DOL_Command_Word_Count RETURN Integer IS
   BEGIN
      RETURN DOL_Command_Word_Count_Val;
   END DOL_Command_Word_Count;

   Function DOL_Command_Footer_Checksum RETURN Integer IS
   BEGIN
      RETURN DOL_Command_Footer_Checksum_Val;
   END DOL_Command_Footer_Checksum;

--------------------------------------------------------------------------------
-- constants used to configure and send DDCS commands in a NASCOM block
--------------------------------------------------------------------------------
```

```
   Function DDCS_Command_Header_SFMDM_Source_ID RETURN Integer IS
   BEGIN
      RETURN DDCS_Command_Header_SFMDM_Source_ID_Val;
   END DDCS_Command_Header_SFMDM_Source_ID;

   Function DDCS_Command_Header_SFMDM_Sync RETURN Integer IS
   BEGIN
      RETURN DDCS_Command_Header_SFMDM_Sync_Val;
   END DDCS_Command_Header_SFMDM_Sync;

   Function DDCS_Command_Header_SFMDM_IOM_Type RETURN Integer IS
   BEGIN
      RETURN DDCS_Command_Header_SFMDM_IOM_Type_Val;
   END DDCS_Command_Header_SFMDM_IOM_Type;

   Function DDCS_Command_Header_DDCS_Command_ID RETURN Integer IS
   BEGIN
      RETURN DDCS_Command_Header_DDCS_Command_ID_Val;
   END DDCS_Command_Header_DDCS_Command_ID;

   Function DDCS_Command_Word_Count RETURN Integer IS
   BEGIN
      RETURN DDCS_Command_Word_Count_Val;
   END DDCS_Command_Word_Count;

   Function DDCS_Command_Footer_Filler RETURN Integer IS
   BEGIN
      RETURN DDCS_Command_Footer_Filler_Val;
   END DDCS_Command_Footer_Filler;

End System_Constants_Package;
```

•

```
-------------------------------------------------------------------------------
-- File Name:       CFGCONST.INI
-- Organization:    NASA - Langley Research Center (LaRC)
-- Project:         LIDAR In-Space Technology Experiment (LITE)
-------------------------------------------------------------------------------
--*****************************************************************************
-- Name/Number:
--    CFG System Constants               (initialization file)
--
-- Abstract:
--    This file contains the values of all of the constants which are accessed
--    through the System_Constants_Package.  They are read in at startup by the
--    Initialize_System_Constants procedure.
--
-- Acronyms/Abbreviations:
--    None
--
-- Dependencies:
--    None
--
-- Global Objects:
--    None
--
-- Exceptions:
--    None
--
-- Machine/Compiler Dependencies:
--    None
--
--*****************************************************************************

-- Normal Color Scheme
[Blue_Background]          -- Background_Color
[Bright_White_Foreground]  -- Foreround_Color
[Yellow_Foreground]        -- Title_Color
[Yellow_Foreground]        -- Border_Color
[Light_Red_Foreground]     -- Highlight_Color

-- Inactive_Color_Scheme
[Blue_Background]          -- Background_Color
[White_Foreground]         -- Foreround_Color
[Yellow_Foreground]        -- Title_Color
[White_Foreground]         -- Border_Color
[Black_Foreground]         -- Highlight_Color

-- Warning Color Scheme
[Red_Background]           -- Background_Color
[Bright_White_Foreground]  -- Foreround_Color
[Yellow_Foreground]        -- Title_Color
[Yellow_Foreground]        -- Border_Color
[Yellow_Foreground]        -- Highlight_Color

-- Standby Warning Color Scheme
[Magenta_Background]       -- Background_Color
[Light_Green_Foreground]   -- Foreround_Color
[Bright_White_Foreground]  -- Title_Color
[Yellow_Foreground]        -- Border_Color
[Yellow_Foreground]        -- Highlight_Color

[Color_80x50]       -- Screen Mode
[80]                -- Number of Columns
[50]                -- Number of Rows
[1]                 -- Max_Top_Row
```

```
[49]                -- Max_Bottom_Row
[79]                -- Max_Column

[8]          -- File Name Length (no extension)
[12]         -- File Name Length (WITH extension (and 'dot'))

[600]        -- NASCOM Block Length

[1]          -- Bit Size
[4]          -- Nibble Size
[16]         -- Word SIze

[15.0]       -- Lookup_Table_1_Minimum_Pressure
[65.0]       -- Lookup_Table_1_Maximum_Pressure

[5.0]        -- Lookup_Table_2_Minimum_Pressure
[65.0]       -- Lookup_Table_2_Maximum_Pressure

[28]         -- Max # of command words in a LITE command block

[22]         -- Max # command words in a LITE command block sent via the PGSC
[30]         -- Total size of LITE command block, including headers

[3]          -- S_Key_Array size
[F1_Key]
[F2_Key]
[F10_Key]

[4]          -- IC_S_Key_Array size
[F1_Key]
[F2_Key]
[F3_Key]
[F10_Key]


[2]          -- size of No_F1_Key_Array
[F2_Key]
[F10_Key]

[1]    -- F10_Key_Array size
[F10_Key]

[ .,<>?/\';:[]{}-_=+|`~!@#$%^&*()]      -- Extra_Chars string for (File ID input)
[80]                                    -- Length of Blanks string

[9]    -- Max number of Time Specific Commands in a LITE command block
[32]   -- Max number of commands, period, in a LITE command block (all Immed, 0 parms)

[24]   -- Command Menu width (for ALL CFG commadn menus)

[3]          -- Menu_Offset
[   ]        -- Menu_Offset_String

[15]         -- Max_Mnemonic_Length
[40]         -- Descriptor_Length
[5]          -- Units_String_Max

[16#50#]     -- Byte_IO_Write_Command_Code
[16#51#]     -- Word_IO_Write_Command_Code
[16#52#]     -- Byte_IO_Read_Command_Code
[16#53#]     -- Word_IO_Read_Command_Code
[16#54#]     -- Memory_Write_Command_Code
[16#55#]     -- Memory_Read_Command_Code
```

```
[16#56#]      -- Memory_Fill_Command_Code

[16#04#]      -- LITE_IC_Null_Cmd_Fill_Pattern_High_Byte
[16#FF#]      -- LITE_IC_Null_Cmd_Fill_Pattern_Low_Byte

[0]    -- Reserved

[5]    -- Show_Menu_Choice_Offset

[28]   -- Max_Memory_Reads
[45]   -- Memory_String_Length
[9]    -- Max_IO_Rows
[5]    -- Max_IO_Cols
[9]    -- IO_String_Length

[8]    -- IC_Command_Frame
[10]   -- IC_Memory_Frame
[53]   -- IC_Uplink_Byte

[16#BEEF#]    -- PDI_Sync_Pattern

[5]    -- Rows_In_Error_Window
[2]    -- Input_Offset_Row
[2]    -- Title_Offset_Row
[4]    -- Bottom_Row_Offset
[6]    -- Yes_No_Box_Rows

[86400]       -- Seconds_Per_Day
[3600]        -- Seconds_Per_Hour

[98]   -- NASCOM_Header_Sync_Byte_1
[118]  -- NASCOM_Header_Sync_Byte_2
[39]   -- NASCOM_Header_Sync_Byte_3
[173]  -- Source_Code
[56]   -- Destination_Code
[7]    -- Block_Sequence_Counter
[9]    -- Format_ID
[255]  -- User_Header_Source_Circuit_ID
[15]   -- User_Header_Source_Circuit_Sequence_Number
[1]    -- User_Header_Spare_Bit
[7]    -- User_Header_Block_Sequence_Number
[143]  -- User_Header_Message_Type
[112]  -- User_Header_Destination_Code
[1]    -- User_Header_Spare_Bit_1
[1]    -- User_Header_Spare_Bit_2
[0]    -- User_Header_Full_Block_Flag
[512]  -- User_Header_Data_Length
[1]    -- NASCOM_Trailer_F1
[0]    -- NASCOM_Trailer_F2
[999]  -- Command_Data_POCC_Command_Message_Number
[55]   -- Command_Data_Payload_Vehicle_ID
[1]    -- Command_Data_Command_Type
[4]    -- Command_Data_Orbiter_Uplink_Mode
[0]    -- Test_Command_Command_Type

[13]   -- LITE_Command_Header_SFMDM_Source_ID
[5]    -- LITE_Command_Header_SFMDM_Sync
[14]   -- LITE_Command_Header_SFMDM_IOM_Type
[3]    -- LITE_Command_Header_SFMDM_Command_ID
[0]    -- LITE_Command_Header_SFMDM_IOM
[0]    -- LITE_Command_Header_SFMDM_Channel_Number
[0]    -- LITE_Command_Header_Fill
[30]   -- LITE_Command_Word_Count
[0]    -- LITE_Command_Footer_Filler
```

```
[13]   -- DOL_Command_Header_SFMDM_Source_ID
[5]    -- DOL_Command_Header_SFMDM_Sync
[13]   -- DOL_Command_Header_SFMDM_IOM_Type
[8]    -- DOL_Command_Header_SFMDM_Command_ID
[0]    -- DOL_Command_Header_SFMDM_IOM
[0]    -- DOL_Command_Header_SFMDM_Channel_Number
[3]    -- DOL_Command_Word_Count
[244]  -- DOL_Command_Footer_Checksum

[13]   -- DDCS_Command_Header_SFMDM_Source_ID
[5]    -- DDCS_Command_Header_SFMDM_Sync
[0]    -- DDCS_Command_Header_SFMDM_IOM_Type
[10]   -- DDCS_Command_Header_DDCS_Command_ID
[8]    -- DDCS_Command_Word_Count
[0]    -- DDCS_Command_Footer_Filler
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)

2. REPORT DATE
October 1998

3. REPORT TYPE AND DATES COVERED
Technical Memorandum

4. TITLE AND SUBTITLE
Designing for Change: Minimizing the Impact of Changing Requirements in the Later Stages of a Spaceflight Software Project

5. FUNDING NUMBERS
WU 258-70-21-10

6. AUTHOR(S)
B. Danette Allen

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
NASA Langley Research Center
Hampton, VA 23681-2199

8. PERFORMING ORGANIZATION REPORT NUMBER
L-17762

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA/TM-1998-208466

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified-Unlimited
Subject Category 61 Distribution: Nonstandard
Availability: NASA CASI (301) 621-0390

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*
In the traditional "waterfall" model of the software project life cycle, the Requirements Phase ends and flows into the Design Phase, which ends and flows into the Development Phase. Unfortunately, the process rarely, if ever, works so smoothly in practice. Instead, software developers often receive new requirements, or modifications to the original requirements, well after the earlier project phases have been completed. In particular, projects with shorter than ideal schedules are highly susceptible to frequent requirements changes, as the software requirements analysis phase is often forced to begin before the overall system requirements and top-level design are complete. This results in later modifications to the software requirements, even though the software design and development phases may be complete. Requirements changes received in the later stages of a software project inevitably lead to modification of existing developed software. Presented here is a series of software design techniques that can greatly reduce the impact of last-minute requirements changes. These techniques were successfully used to add built-in flexibility to two complex software systems in which the requirements were expected to (and did) change frequently. These large, real-time systems were developed at NASA Langley Research Center (LaRC) to test and control the Lidar In-Space Technology Experiment (LITE) instrument which flew aboard the space shuttle Discovery as the primary payload on the STS-64 mission.

14. SUBJECT TERMS
software design, change

15. NUMBER OF PAGES
121

16. PRICE CODE
A06

17. SECURITY CLASSIFICATION OF REPORT
Unclassified

18. SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

19. SECURITY CLASSIFICATION OF ABSTRACT
Unclassified

20. LIMITATION OF ABSTRACT