NASA TM-81924

NASA Technical Memorandum 81924

NASA-TM-81924 19810010177

AN EVALUATION OF THE INTERACTIVE SOFTWARE INVOCATION SYSTEM (ISIS) FOR SOFTWARE DEVELOPMENT APPLICATIONS

MARIE S. NOLAND

JANUARY 1981

National Aeronautics and Space Administration

Langley Research Center
Hampton, Virginia 23665

SUMMARY

The Langley Research Center (LaRC) project called Multipurpose User-Oriented Software Technology (MUST) is intended to reduce the cost of producing flight software and to provide an integrated system of support software tools for flight projects. One of the tools developed under this project was the Interactive Software Invocation System (ISIS) which allows a user to build, to modify, to control, and to process a total flight software system without direct communications with the host computer. ISIS was developed primarily for the handling of large amounts of information that are typically required to support flight projects. In order to evaluate the effectiveness of the system as applied to a flight project, the collection of software components required to support the LaRC Annular Suspension and Pointing System (ASPS) flight project were integrated using ISIS. The ASPS project information was organized under ISIS according to the software development life cycle concept. This resulted in an integrated system adequate to perform the verification and validation requirements for the ASPS flight software code. This system contains an ordered collection of information consisting of language processors, automated documentation and design tools, flight software utility routines, simulation subsystems, and testing procedures which are controlled through the interactive user interface capability provided by ISIS.

This report discusses the ASPS software system, the ISIS features, the organization of an ISIS library, and the integration of ASPS into ISIS. The 5-level hierarchical file structure of ISIS proved to be very useful in organizing the ASPS flight software information in a more meaningful manner than allowed with the Network Operating System (NOS), but ISIS proved to be less efficient than NOS in the storing and retrieving of files. The Interactive Programming Language (IPL) capability is one of the strongest features of ISIS but one that requires a great deal of effort by a user to exercise it to the fullest extent possible. ISIS is still in the development phase with several improvements planned to enhance its value and usability.

INTRODUCTION

The Langley Research Center Multipurpose User-Oriented Software Technology (MUST) project was designed to reduce the cost of producing software for digital systems used in flight research by providing an integrated system of software tools for use throughout the flight software development process. The Integrated Software Invocation System (ISIS), one of the MUST tools, is an

N81-18703#

interactive data management system providing the user with a file manager, text editor, a tool invoker, and an Interactive Programming Language (IPL). The basic file design of ISIS is a 5-level hierarchical structure. The file manager controls this hierarchical file structure and permits the user to create, to save, to access, and to purge pages of information. The text editor is used to manipulate pages of text to be modified and the tool invoker allows the user to communicate with the host computer through a RUN file created by the user. The IPL is based on PASCAL and contains most of the statements found in a high-level programming language. The IPL program is used to provide an interactive interface between ISIS and the user, permitting a user to interact with ISIS through a prompting process provided by the IPL.

The Annular Suspension and Pointing System (ASPS) is under development to provide the improved pointing capability required for the Shuttle sortie missions. To support the software development of ASPS requires large quantities of information consisting of documentation, design and implementation tools, and testing procedures. Determining how best to use and control this vast amount of information led to evaluating the utility of ISIS and other software support tools developed under MUST as applied to flight projects.

DISCUSSION OF THE ASPS SOFTWARE DEVELOPMENT

The software development life cycle concept was followed in the organization of files for this flight project. The logical groupings for the various pieces of information in the ASPS flight system are as follows: documentation, design tools, implementation tools, and testing tools.

The ASPS flight software documentation requirements consist of a Software Design Document (SDD), Software Standards Document (SSD), Software Requirements Document (SRD), General Software Development Test Plan (SDTP), and Program Specifications Document (PSD).

Two design tools used in the development and debugging phases of the ASPS are a text processor and a structured flowcharter. The RNF text processor is a text formatting system used to create machine acceptable text and to produce a readable version of documentation in upper and lower case. It accepts free-format text files and produces output suitable for a terminal, a line printer, or a typewriter. The structured flowcharter is a flow diagramming tool for documenting and understanding programs. It produces flowcharts for programs written in the high-level language HAL/S. The flowcharts are presented in a top-down flow that can be used in the design, implementation, and debug phases of flight software. This system can be used to produce flowcharts on either a printer or a plotter.

The implementation tools consist of assemblers, loaders, compilers, and cross-compilers. A META ASSEMBLER was used in the testing and the application of the ASPS simulators and in the assembly of the actual ASPS flight code to produce an object module for processing. The object module is generated according to the machine definition for the ASPS flight computer which is input to the META ASSEMBLER. This code is input to a Linkage Editor creating a load module which may be executed on the target computer or by a simulator. The HAL/S cross-compiler and the ASPS simulators are written in PASCAL.

The testing tools used for ASPS include simulators, a utility library, and test routines. The flight computer supporting the ASPS was developed by IBM essentially as a flight qualified version of IBM System 360 and is called the NASA Standard Spacecraft Computer-II (NSSC-II). Since the NSSC-II computer was not available for the software development at LaRC, a NSSC-II Interpretive Computer Simulator (ICS) was developed to perform the verification and validation of the actual ASPS flight software. Currently, the ASPS flight code consists of a Real-Time Executive Module (RTEM) written in assembly language and an Attitude Control Module (ACM) and an Attitude Determination Module (ADM) written in HAL/S. Another of the testing tools is a closed-loop system simulation consisting of the NSSC-II ICS and an ASPS Gimbal System (AGS) simulation. The simulation system also includes a utility support library, test routines, and test procedures necessary to assemble, compile, link and load, and execute the ASPS flight software code.

DISCUSSION OF ISIS

ISIS is an interactive data management system which provides such capabilities as a file manager, text editor, and a tool invoker, each of which may be controlled by a PASCAL-based Interactive Programming Language (IPL). The file manager allows the user to access pages within a 5-level hierarchical structured file system. The text editor manipulates pages of text contained in the file system and the tool invoker allows communications with the resident operating system. An IPL program includes statements for text editing, file management, and tools invocation.

ISIS was developed for the MUST project by Dr. W. Joseph Berman of the University of Virginia. This system was designed for machine independence and currently resides on the Langley Research Center CYBER series computers running under the Network Operating System (NOS). For an in-depth discussion of ISIS refer to the ISIS USERS MANUAL (ref. 1).

File Manager

The ISIS file manager controls access to a 5-level hierarchical file structure as shown in figure 1. The highest level of this structure is the library. The library contains shelves, a shelf contains books, a book contains chapters, and a chapter contains pages. A NOS file is stored at the page level, the other four levels are used for naming convention only, allowing the user to identify each page of information. Pages may contain various types of information such as documentation, source code, binary code, data, or control cards. All 5 levels of the file structure must be specified when first entering the library. The file structure is written in the form:

library.shelf.book.chapter.page

The user can access, save, and purge pages of information in the library through ISIS control statements. The USE statement reads the contents of a specified page from the library into a specified work space called a frame for additional processing. The name of this page may be changed with the SET NAME statement. The new page name can be saved or replaced with the SAVE or SAVE* statement. The PURGE statement eliminates the specified page from the library. If the specified page is the only page in the chapter, the chapter will be eliminated. This process continues through the book and shelf level and an empty library will remain. The library is never eliminated in this process, it must be purged from the system by the user.

Text Editor

The ISIS editor is line oriented and not pointer oriented as are many text editors, therefore each line of text must be uniquely identified. The ISIS editor does this by assigning line numbers to each line of text. These numbers represent reference points for the specification of text by the editor. The text is referenced by designating the line of text to be modified within each command. The lines affected by the edit command are referred to as the RANGE of the command which can be explicit or implicit. The explicit range permits the user to operate on everything within a certain area of the page. The implicit range specifies a search of all lines in the text having a particular characteristic. The implicit and explicit ranges can be combined, such as a particular characteristic between certain line numbers in the page.

During editing a working frame is used as temporary storage. The frame can be a copy of an existing library page to be modified or it can be a new page entered by the user. There are 12 working frames available; the two frames WORK and SHOWN are provided and named by the system, the other 10 frames must be named by the user. The ISIS editor permits the user to edit multiple frames simultaneously. The user can declare a working frame with the FRAME statement. This frame can then be declared ACTIVE and be viewed in all or part with the LIST command. Text to be stored in this frame can be entered from the terminal with the INSERT command. The CHANGE command will modify existing lines of text in the frame. The range of the CHANGE command may consist of an implicit or explicit range and specify the change be made in a certain column. The EXEC command permits the user to execute the contents of the ACTIVE frame. Portions of this frame can be executed by using the range option. Since frames are temporary storage, it is necessary to SAVE the frame as a page in an ISIS library using the ISIS file manager.

The interrogation statements give the user access to system and programming information during an interactive session. For example, SHOW RESERVED will display the ISIS reserved words and SHOW ABBREVS will display the current user declared abbreviations.

Tool Invocation Statements

The tool invoker is made up of three statements. The RUN statement causes information to be concatenated in a special sequence to form an acceptable INPUT file that can be submitted to the host computer. This INPUT file contains control cards, source programs, and/or data with a right parenthesis [)] used to separate each record. The file is referred to as a RUN file in ISIS and RFILE in NOS. The SEND statement allows the user to submit the RUN file to the NOS batch system. The STOP:SEND statement is used for an interactive session. The STOP statement terminates an ISIS session and the SEND statement submits the RUN file to the host operating system interactively.

Interactive Programming Language

The Interactive Programming Language (IPL) is based on the high-level language PASCAL. The user communicates with ISIS through "commands", these commands can be user activated or activated through an IPL program. A command is defined as a sequence of one or more statements with the number of commands in an IPL program depending on the user's needs. The statements can be entered at the terminal and executed as a one time only interactive session or stored as a page in ISIS for repeated executions. The user can write an IPL program to build a set of control cards, another IPL program to submit the control cards to NOS with specified inputs, and still another program to edit the control cards. Without the IPL programs the user would build the control cards with the INSERT statement, edit them with the editing statements (CHANGE, DELETE, MODIFY, ADD), and send them to NOS with the RUN and SEND statements.

The IPL differs from PASCAL in that it does not require a BEGIN and END to surround compound statements (all that is required is an END statement for termination of compound statements) nor does it allow array packing, CASE statements, or subrange types. The IPL allows some PASCAL data types in an abbreviated form such as INT and BOOL. ISIS includes two data types not in PASCAL: STRING and KEY. The STRING contains alphanumeric information enclosed in quotation marks. KEY is defined as a line number assigned to each line of text. The line number can be any type of KEY operand, a KEY variable or expression, or a simple number. The KEY type permits the user to define variables to be used in the range of the edit commands.

The two groups of programming statements are the Declarative statements describing the program variables, and the Action statements which are the executable statements. The Declarative statements are ABBREV, TYPE, VAR and ERASE. The ABBREV statement permits the user to abbreviate the ISIS statement verbs. The TYPE and VAR statements are similiar to the TYPE and VAR statements of PASCAL which declare various data structures and variables, respectively. If the user fails to declare all variables being used, instead of aborting the command, the ISIS system will interrogate the user. The ERASE statement eliminates the specified types and variables from the identifier table. The Action statements include an assignment statement and control statements such as LOOP, REPEAT UNTIL, WHILE, FOR, and IF-THEN-ELSE. The compile and execute statements include XEQ and EXEC. XEQ takes the contents of a string expression and interprets it as a command to the system, the EXEC statement allows the user to execute an ACTIVE or specified frame. The ASK, PRINT and PRINTLN statements are interactive terminal input/output statements. The ASK statement will interrupt program processing and prompt for terminal input. The PRINT and PRINTLN statements print terminal output.

APPLYING ISIS TO ASPS

The purpose of this case study was to assemble an integrated system of existing software tools using the Interactive Software Invocation System (ISIS) for the support of the Annular Suspension and Pointing System (ASPS) software development. The number of files (documentation, source, binary, data, and test case) required to support the ASPS flight project is large. These files are stored in NOS as shown in figure 2 with no special organization. ISIS gives the user the capability of organizing these files in a library. The organization of a library is a task requiring a great deal of thought. For example, it was decided to organize the ASPS files according to the software development life cycle concept. One alternative to this would be to group files along functional lines. Files should be grouped into proper categories and each category should have a meaningful name. In the 5-level hierarchical file structure all information is stored at the page level and the remaining four levels are used for naming convention only. Each level name has a maximum of seven characters.

Library For ASPS Tools

The ASPS library organization is based on four software development life cycle categories: documentation, design tools, implementation or coding, and testing. These four categories became the shelf labels for the ASPS library ASPSLIB as shown in figure 3. The documentation (DOCMNT) shelf contains books labelled Software Design Document (SDD), Software Standards Document (SSD), Software Development Test Plan Document (SDTP), and Software Requirements Document (SRD). The SRD book contains chapters of actual flight software modules labeled Real-Time Executive Module (RTEM), Attitude Control Module (ACM), and Attitude Determination Module (ADM). The RTEM chapter contains pages labeled Table of Contents (TC), section I (I), and so forth. These pages contain the actual information stored into this library. The tools (TOOLS) shelf contains a similar type organization for the design tools with books labeled Flowcharter (FLOWCHT) and Wordprocessor (WRDPROC). The FLOWCHT book contains chapters labeled HALS programs (HALS), control cards (CONTROL), input (INPT), and information (INFO). The CONTROL chapter contains pages labeled HALCAL, HALPRTR, and HOS. The HALCAL page contains the control cards necessary to produce a CALCOMP flowchart, the HALPRTR page contains the control cards necessary to produce a printed flowchart, and the HOS page contains the necessary control cards to compile the flowcharter source program. This type of organization can be applied to any project and can consist of a larger or smaller number of shelves, books, chapters, and pages. Figure 4 shows an example of a library layout for the Terminal Configured Vehicle (TCV) flight project based on a functional organization. The flight project was organized into the three functional processes: flight controls, navigation and guidance, and displays labelled FLIGHTCTL, NAVGUID, and DISPLAYS, respectively. These three functional processes form the shelf level. The book level requires the same type, but distinct, books for each shelf. That is, each shelf requires its own REQUIREMENTS book, its own TEST PLANS book, and so forth. The REQUIREMENTS book contains the AUTOLAND chapter. The AUTOLAND chapter contains the LATERAL, LONG, FLARE, and ROLLOUT pages.

Figures 5 through 7 show the layout of the library ASPSLIB. The design tools were organized into a TOOLS shelf as shown in figure 6. This shelf (TOOLS) will be used to show the process required for a library organization. The TOOLS shelf contains three books, the first book contains all pages associated with the flowcharter, the second book contains all pages associated with the RNF text processor, and the third book contains general information about the TOOLS shelf. The flowcharter book contains four chapters. The first chapter contains pages of flowcharter source, binary, and HAL/S grammar to flowchart a HAL/S program. The second chapter contains pages of control cards required to execute the flowcharter and produce plots in printed or plotted form. The third chapter contains pages of input (HAL/S programs), and the fourth chapter contains descriptive information about the contents of the FLOWCHT book on this shelf. The page level contains the actual files as they were originally stored on NOS. These files include source programs for the flowcharter, binary files for the flowcharter and RNF, a HAL/S grammar for the flowcharter, input files for the flowcharter and RNF, output files for RNF, and control cards required to execute the flowcharter and RNF.

Library Directory Utility

In addition to the four shelves described above, ASPSLIB contains an additional shelf called GENERAL containing information about the contents of ASPSLIB. This library directory was built for the purpose of guiding a user through the ASPSLIB library and it is stored as pages in ASPSLIB. Figure 8 shows the ISIS data base for ASPSLIB and the arrows indicate the direction of a particular library walkthrough. Figures 9 through 13 show the information contained in the directory at each level of the library for this example and the steps necessary to move through each level. Within the library ASPSLIB, the DOCMNT shelf was selected for this example walkthrough, from this shelf the SRD book was selected, from this book the RTEM chapter was selected, and from this chapter the II page was selected. The actual library walkthrough begins with an information file at the library level as shown in figure 9. This information file can guide a user to any part of the library without having to know all the ISIS commands. The first entry into the library requires that all 5-levels be specified. The ISIS command USE reads the contents of the "file" page into an active frame. The LIST:NK lists the contents of the active frame without line numbers (KEYS) to the left of the text. The contents of this frame gives the user a brief description of the library, shows the shelf levels available and tells how to access the shelf of interest. The DOCMNT shelf was chosen as the next area of interest in this walkthrough. The next step is the shelf level as shown in figure 10. To access this level the user only needs to specify 4 levels. The contents of this frame describes the books available on this shelf. The SRD book was chosen as the next area of interest. To access the book level the user only needs to specify 3 levels as shown in figure 11. The contents of this frame describes the chapters available in this book. The RTEM chapter was chosen as the next area of interest. To access the chapter level the user only needs to specify 2 levels as shown in figure 12. The contents of this frame describes the pages available in this chapter. The II page was chosen as the next area of interest and is shown in figure 13. The page level contains the actual page (file) of information stored in the library.

Submit File Utilities

Figure 14 shows a specific ISIS submit file contained in ASPSLIB which differs from a NOS submit file in the following ways. For ISIS the /JOB has been deleted and the EOF has been replaced by a right parenthesis [)]. The IFETCH and ISISGET are required by ISIS to retrieve binary files and source files, respectively. Each file retrieved from ISIS must be specified in this manner. Even if two pages are in the same chapter all 5 levels must be specified. This submit file will compile and execute the specific input file (ASPS1) specified in lines 9.1 and 13. The ISTORE and ISISPUT stores or replaces binary files and source files, respectively. The RUN command creates an input file (control cards and source program) to submit to NOS. The KEYS or line numbers are removed for NOS submittal using the no key (NK) parameter with the RUN command. The SEND command submits this RUN file to NOS.

An ISIS IPL program can be very useful in accessing pages from the file manager, with the capability of editing and replacing these pages, then writing the control cards necessary to submit this information to the host computer for execution. Figures 15 through 17 show an IPL program built to execute a general purpose submit file, the general purpose submit file to be executed, and the prompting process for executing this IPL program. Figure 15 shows an IPL program that erases and assigns frames and variables, prompts the user for another job, and clears the RUN file. The user must specify the submit file needed for a general purpose submit. The program will prompt the user for the source page (input) name. When the source page has been assigned the program will ASK if you are ready to SEND. The IPL program will then loop and prompt for another submission. The general purpose submit file as shown in figure 16 is the submit file from ISIS shown in figure 14 edited for a variable input file. The ASPS1 page was not retrieved nor was it specified as input on the PASCAL line. This submit file was built for variable inputs but only one input file is permitted for each submission. Execution of an IPL program showing the actual prompting process of an IPL program is shown in figure 17. The lower case type in figure 17 are the user's response to the IPL prompts. The program prompts the user for a job, clears the RUN file, inserts the submit file requested (as shown in figure 16), requests the source page name, shows the number of lines inserted, and ASKs if the user is ready to SEND the file to NOS.

NOS files must be edited and submitted manually. Retrieving and storing files in the NOS system requires less effort on the part of the user and currently is much more efficient than ISIS. Some differences in the ISIS text editor and the NOS text editor are noteworthy. The ISIS text editor is line oriented and the NOS editor is pointer oriented. The commands in both editors can be abbreviated requiring fewer key strokes for the user. The abbreviations are the default in NOS, in ISIS the user must define the abbreviations and these abbreviations can be geared to each user's wishes. The NOS editor can create or process an ASCII (upper and lower case) file. The ISIS editor has limited capability in this area. The NOS editor automatically changes all occurrences on the line of text, ISIS changes only the first occurrence unless the multiple occurrence option is selected. In ISIS, KEYS are required for editing and are shown in figure 18 with the smallest possible increment.

CONCLUDING REMARKS

This case study resulted in an organized file system containing existing flight software development and support tools. This system is an integrated file system capable of supporting the verification and validation requirements of the ASPS flight code. The 5-level hierarchical file structure of ISIS is extremely useful in identifying and ordering pieces of information to be stored at each level. The capability of organizing files in this manner can be more valuable to a user than the Network Operating System (NOS) organization for large amounts of information. Files are stored randomly in a NOS catalog as shown in figure 2; whereas, in ISIS the organization is controlled by the user as shown in figure 3.

This paper reflects the state of ISIS at the time the case study was conducted in mid 1980. Since ISIS is still in the development and implementation phase, several desirable features are planned to be installed as demand requires and time allows. For example, procedures, functions, and a directory editing capability are presently being implemented. The directory editing is necessary for the editing of a library layout. Currently, if an error is made in any of the names while storing pages into the library this erroneous name will remain in the library until the user purges this page and stores it again. Also, when a new page is stored into the library this new page automatically falls to the bottom of the chapter level. At present there is no capability for storing a page in a particular location. After the initial library in ISIS has been built if the user wishes to change the name of a page or store several additional pages, the ISIS library layout begins to look as though it has the same type of organization as a NOS catalog. With directory editing this could be avoided and the user would have the capability of storing a page of information at any level in the library. This editing capability should enable the user to have a clean and well organized library at all times.

An IPL program can be built that will control the file manager, text editor, and tool invoker. This allows the user to access a page of information from the file manager, edit and replace this page, and then submit it to the host computer through an IPL program. This is an advantage over the use of a set of independent tools required to perform the same task. The IPL programs must be built by someone with considerable knowledge of ISIS, the host operating system, the use of terminals, and the PASCAL language. If an IPL program has been built for a user by an ISIS system builder then the user is only required to know how to use a terminal and have minimum knowledge of the host operating system to perform the same task.

In conclusion the 5-level hierarchical file structure of ISIS was extremely useful in organizing the software components required for the ASPS flight project. This type of organization proved to be more meaningful than the NOS organization, but in the storing and/or retrieving of files ISIS proved to be less efficient than NOS. One of the strongest features of ISIS is the IPL capability, but it is one that requires a great deal of effort and system knowledge by the user to exercise to the fullest extent. ISIS is still in the development and implementation phase with several improvements planned to enhance its usability.

Langley Research Center
National Aeronautics and Space Administration
Hampton, VA 23665
December, 1980

REFERENCES

1. Grantham, Carolyn: ISIS Users Manual
NASA Technical Memorandum 80144, March 1980.

BIBLIOGRAPHY

1. Cormack, A., III; Andrews, P. D.:
Flight Experiment Definition of an
Annular Suspension and Pointing System. (ASPS),
Rockwell International,
SD 77-AP-0027, May 1977

2. Berman, W. Joseph : Functional Description of the ISIS System.
NASA CR-159152, October,1979

3. Straeter, Terry A.; Foudriat, Edwin C.; Will, Ralph W.:
Research Flight Software Engineering and MUST,
an Integrated System of Support Tools.
Proceedings COMPSAC77 (IEEE 77CH1291-4C), pp.392-396, Nov. 1977

4. Zelkowitz, Marvin V.:
Perspectives on Software Engineering.
Computing Surveys, Vol. 10, No. 2, June, 1978, pp. 197-216

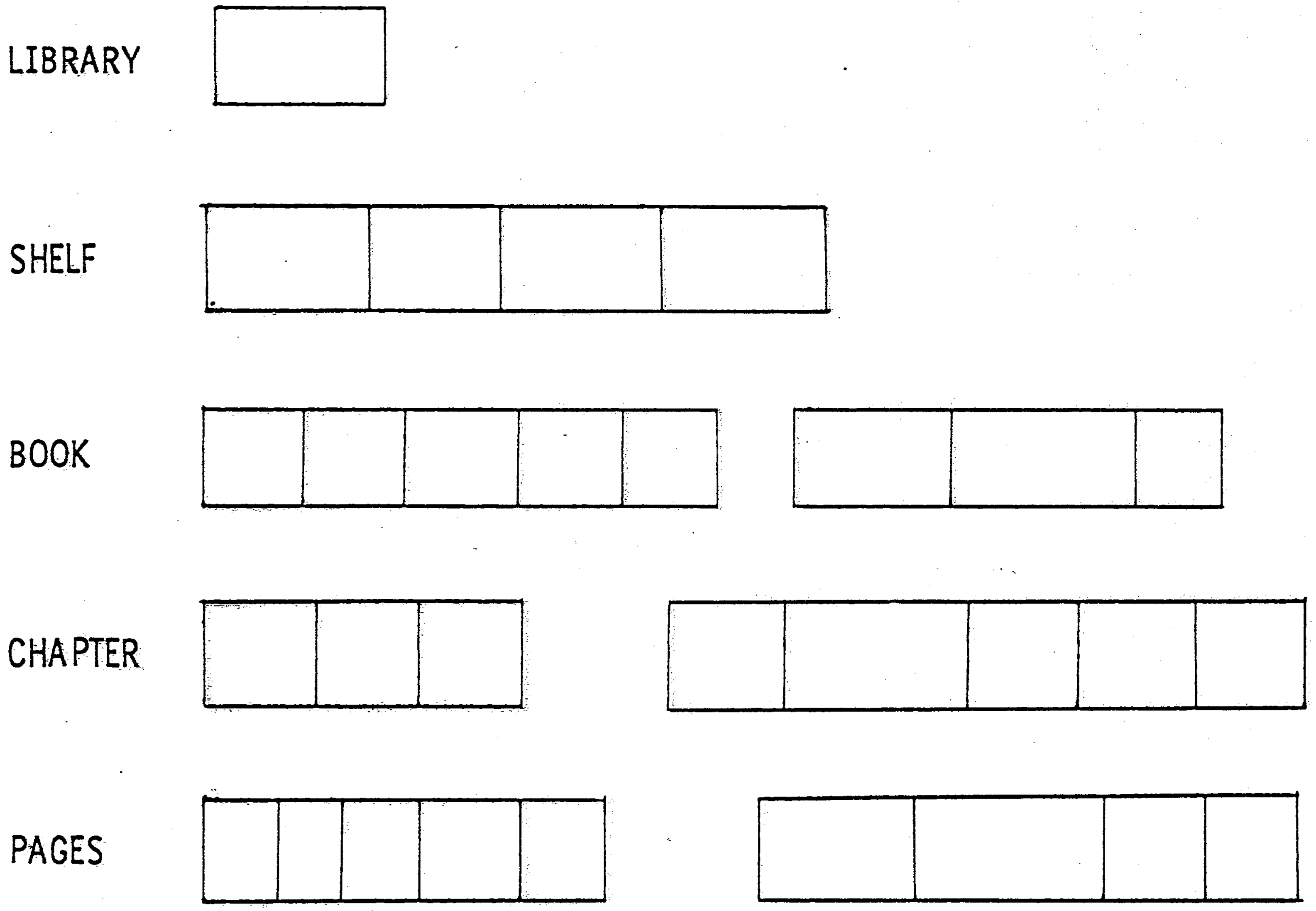

Figure 1. – The 5-level hierarchical file structure

```
PLTREXT   SQRTFL    HOSFCPR   TESTJRW   PRTRLGO   ASPS1     PLTRLGO
CONTRLF   HGTAB     DATAF     HALCAL    ASPSCL    HALPRTR   LGO
HOS       EXEC      HALMOD    LOADMOD   ROMAN     SIMFLT    RNFPRTR
SIMFLTM   RNFTLL    TESTSIM   AGSSRD    SIMLAT    FINAL     NSSCII
SFILE     NSSCII1   ICSLIB    SIMEXEC   AGSCL     MACLIB    METAMOD
MODBIN    TAPE15    CONIN     LKEDMOD   LKEXAB    NSCLIB    LIBTP
TAPE8     FLOATPT   SQRTS     TEST      FLOATPM

       47 FILE(S)
```

Figure 2. – Listing of ASPS files as stored under NOS

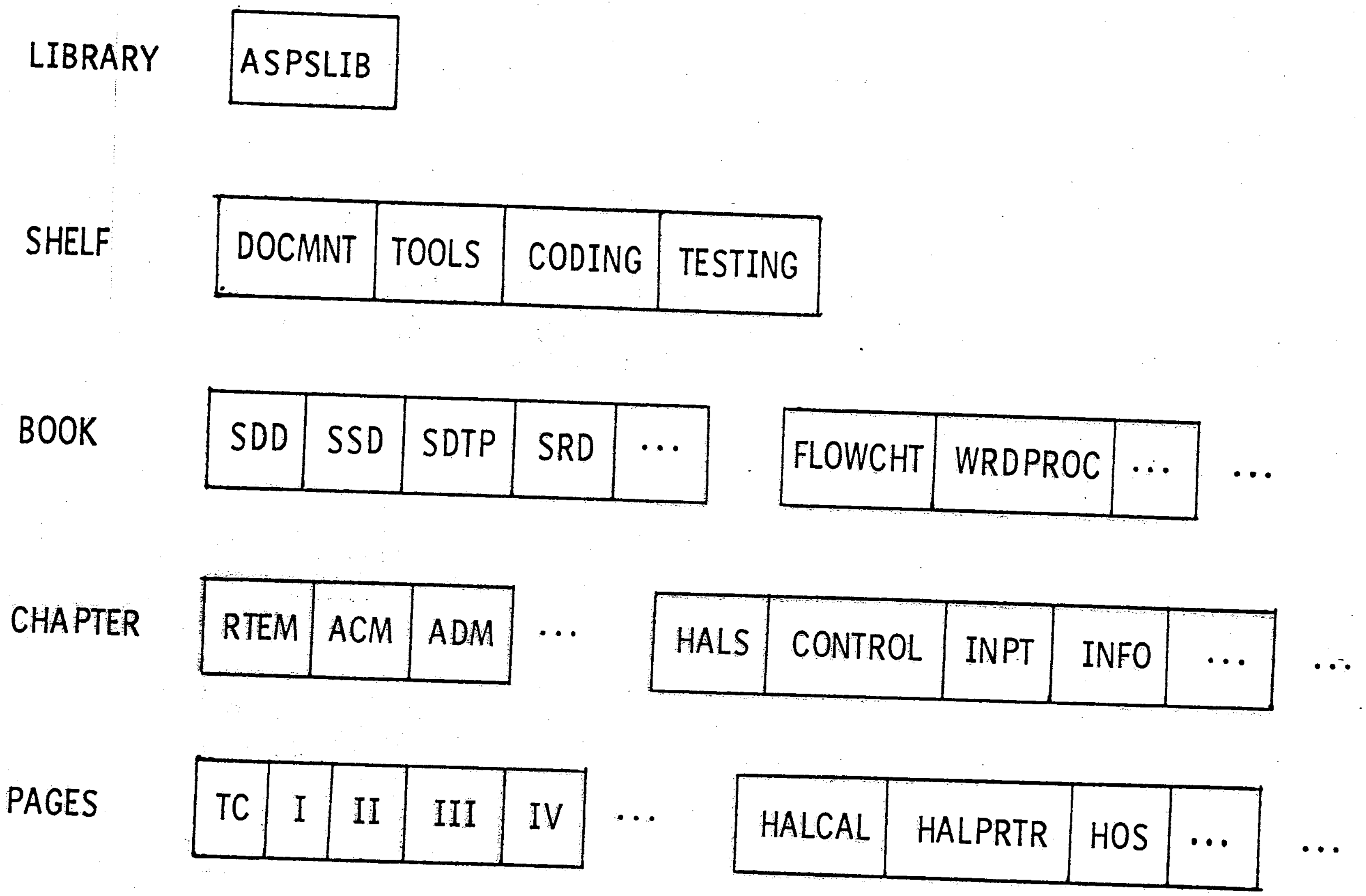

Figure 3. - Example ISIS data base for ASPS flight project

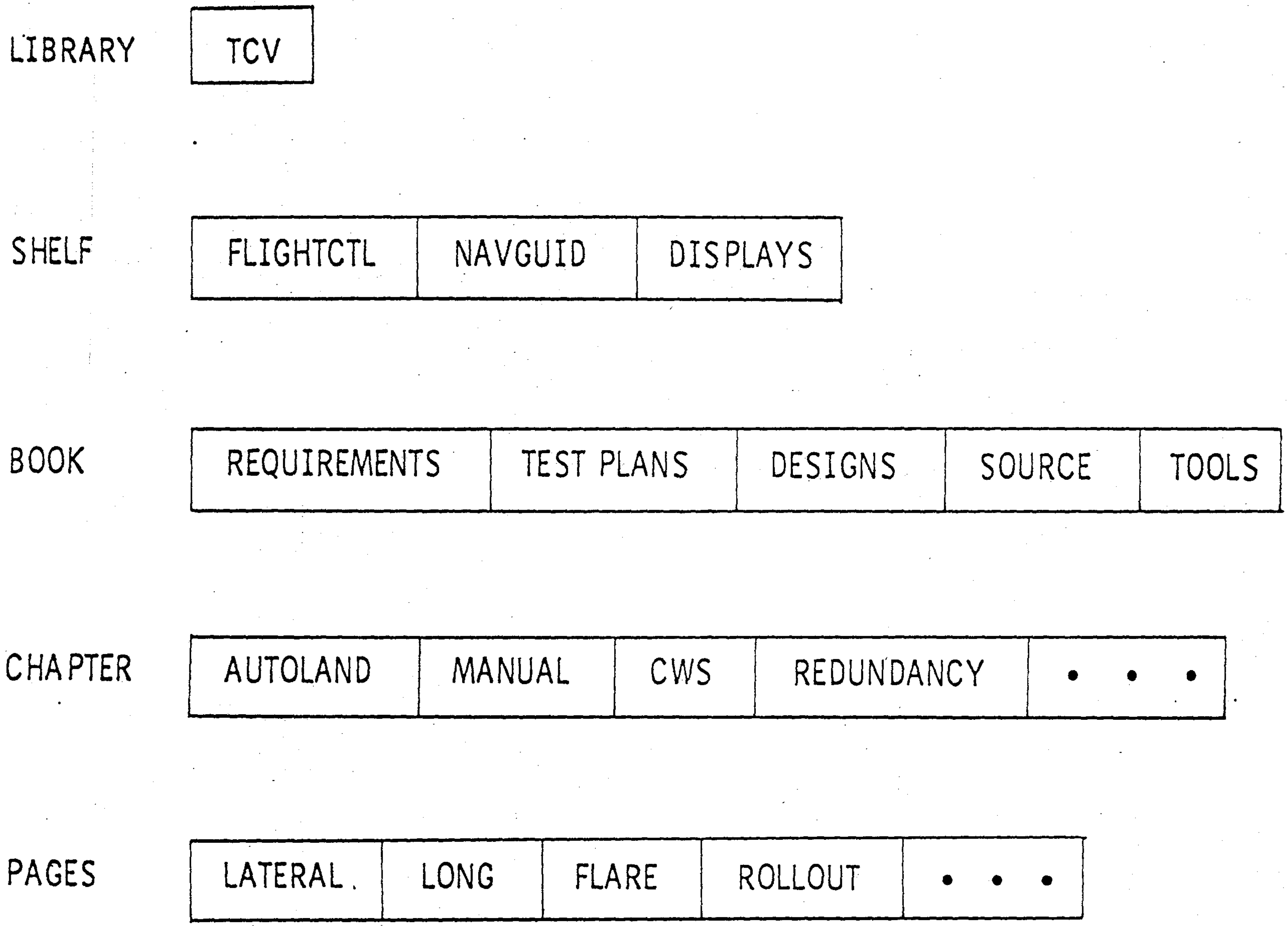

Figure 4. - Example ISIS data base for TCV flight project

show pages.

```
ASPSLIB.DOCMNT .SDD      .REF      .TC
        .       .         .        .SDD3
        .       .         .        .PRINT
        .      .SSD      .REF      .TC
        .       .         .        .I
        .       .         .        .II
        .       .         .        .III
        .       .         .        .IV
        .       .         .        .V
        .       .         .        .TC1
        .      .SDTP     .REF      .TC
        .       .         .        .I
        .       .         .        .II
        .       .         .        .III
        .       .         .        .IV
        .       .         .        .V
        .      .SRD      .REF      .AGSSRD
        .       .        .RTEM     .TC
        .       .         .        .I
        .       .         .        .II
        .       .         .        .III
        .       .         .        .IV
        .       .        .ACM      .TC
        .       .         .        .I
        .       .         .        .II
        .       .         .        .III
        .       .         .        .IV
        .       .         .        .V
        .       .         .        .VI
        .       .        .ADM      .TC
        .       .         .        .I
        .       .         .        .I1
        .       .         .        .I2
        .       .         .        .II
        .       .         .        .II1
        .       .         .        .II2
        .       .         .        .II3
        .       .         .        .II4
        .       .         .        .III
        .       .         .        .IV
        .       .        .INFO     .TC
```

Figure 5. - ASPSLIB library layout

```
ASPSLIB.TOOLS  .FLOWCHT.HALS    .PLTREXT
       .       .       .        .HOSFCPR
       .       .       .        .PRTRLGO
       .       .       .        .PLTRLGO
       .       .       .        .HGTAB
       .       .       .        .PLTRLG3
       .       .       .CONTROL.HALCAL
       .       .       .        .HALPRTR
       .       .       .        .EXEC
       .       .       .        .HOS
       .       .       .        .TC
       .       .       .INPT    .HALMOD
       .       .       .        .ROMAN
       .       .       .INFO    .TC
       .       .WRDPROC.AIDS    .EXEC
       .       .       .        .RNFHELP
       .       .       .        .RNFMAC
       .       .       .        .RNFHF
       .       .       .ASCIIPR.RNF
       .       .       .        .RNFTL
       .       .       .CONTROL.RNFPRTR
       .       .       .        .RNFTLL
       .       .       .        .EXEC
       .       .       .        .TC
       .       .       .INPT    .AGSSRD
       .       .       .OTPUT   .FINAL
       .       .       .        .SFILE
       .       .       .INFO    .TC
       .       .GENERAL.INFO    .TC
       .CODING .METAASM.MCDAC   .ICSLIB
       .       .       .        .SFMETA
       .       .       .        .MACLIB
       .       .       .        .METAMOD
       .       .       .        .MODBIN
       .       .       .        .TAPE15
       .       .LOADER .LINKED  .CONIN
       .       .       .        .CONINEX
       .       .       .        .LKEDMOD
       .       .       .        .LKEXAB
       .       .       .        .NSCLIB
       .       .       .        .LIBTP
       .       .       .        .TAPE8
       .       .GENERAL.INFO    .TC
```

Figure 6. – ASPSLIB library layout (continued)

```
ASPSLIB.TESTING.ICS     .MODULE .FLOATPT
        .       .       .       .SQRTS
        .       .       .       .TEST
        .       .       .       .FLOATPM
        .       .       .       .SQRTFL
        .       .       .       .TESTJRW
        .       .       .INTEGRT.ASPS1
        .       .       .       .CONTRLF
        .       .       .       .DATAF
        .       .       .       .ASPSCL
        .       .       .       .CKPTF
        .       .       .       .LGO
        .       .       .       .OLLGO
        .       .       .       .DATAF3
        .       .       .       .LGOO
        .       .       .       .LGOG
        .       .       .       .LGOAGS
        .       .       .       .DATAF2
        .       .SYSTEM .SOFTWR .EXEC
        .       .       .       .LOADMOD
        .       .       .       .DUMPP
        .       .       .       .LOADMDP
        .       .       .       .TAPCONV
        .       .       .       .FILE2
        .       .       .HARDWR .RSAGS
        .       .       .CONTROL.SIMFLT
        .       .       .       .SIMFLTM
        .       .       .       .TESTSIM
        .       .       .       .SIMLAT
        .       .       .       .NSSCII
        .       .       .       .NSSCII1
        .       .       .       .SIMEXEC
        .       .       .       .AGSCL
        .       .       .       .CKPTDMP
        .       .       .       .SPERRY1
        .       .       .       .SPERRY2
        .       .       .       .SPERRYT
        .       .       .       .SIMFLTT
        .       .       .       .SIMTEST
        .       .       .       .NSSCI
        .       .       .NSSCII .EXEC
        .       .       .       .RITSUB
        .       .GENERAL.INFO   .TC
```

Figure 7. – ASPSLIB library layout (concluded)

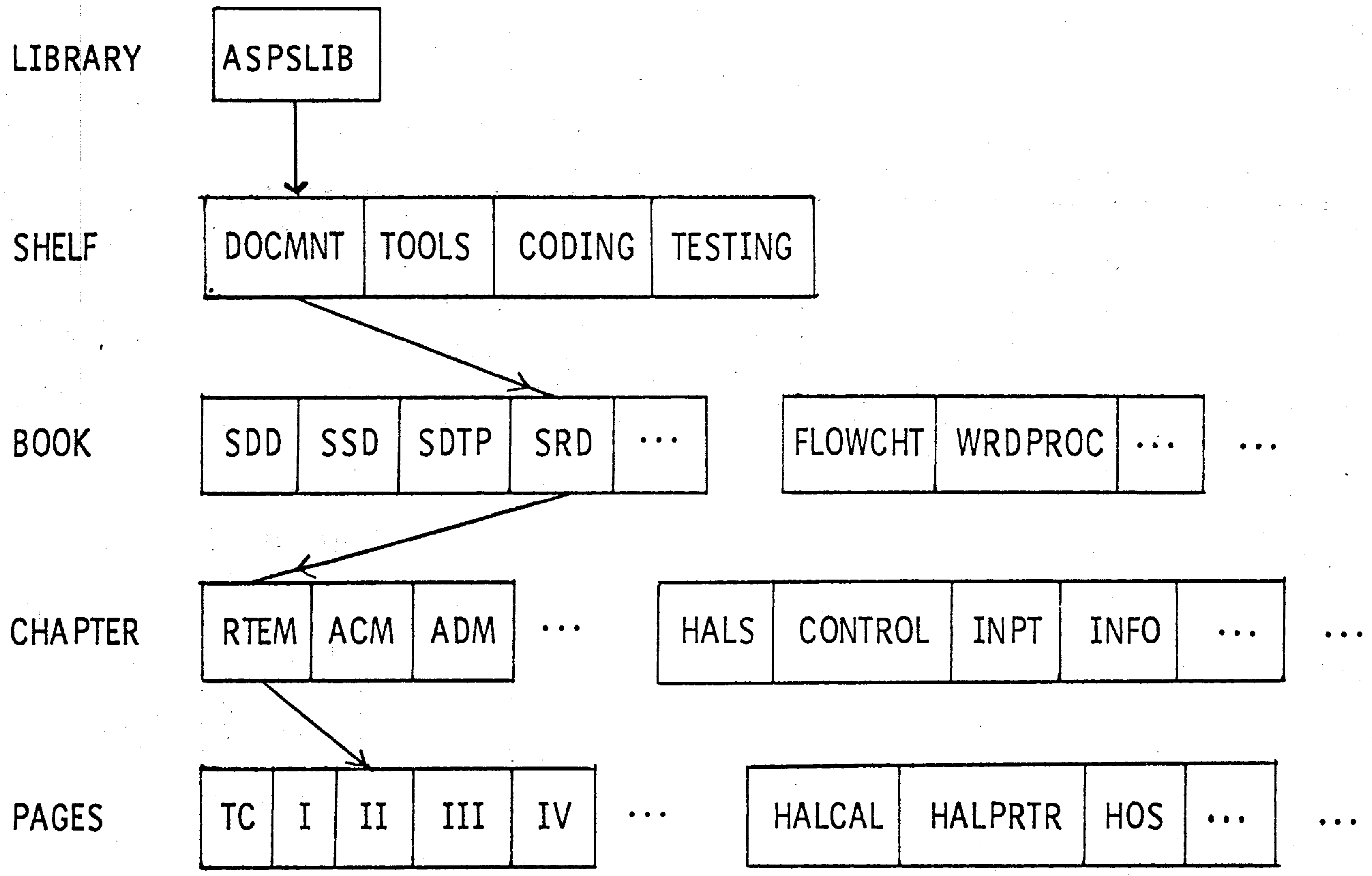

Figure 8. – Direction of library walkthrough

```
        use aspslib.general.library.info.file;list:nk

WORK         USED FROM ASPSLIB.GENERAL.LIBRARY.INFO.FILE
1

                                                                    1


This library consists of a collection of files required to
support the ASPS flight project.  These files include ASPS
documentation, design tools, coding aids and testing systems.  To
access the area of interest enter one of the following commands.

        USE DOCMNT.GENERAL.INFO.TC;LIST:NK

        USE TOOLS.GENERAL.INFO.TC;LIST:NK

        USE CODING.GENERAL.INFO.TC;LIST:NK

        USE TESTING.GENERAL.INFO.TC;LIST:NK


                              NOTE

The above commands must be explicitly designated to move from one
area of the library to another.  Take note as this will be the
ONLY appearance of these commands.

To exit ISIS enter STOP
```

Figure 9. – Library level

```
          use docmnt.general.info.tc;list:nk

WORK       USED FROM ASPSLIB.DOCMNT.GENERAL.INFO.TC
1

                                                                   1


 This shelf contains information on ASPS documentation.  To access
 the area of interest enter one of the following commands.

                   (for Software Design Document)
                       USE SDD.REF.TC;LIST:NK

                  (for Software Standards Document)
                       USE SSD.REF.TC;LIST:NK

             (for General Software Development Test Plan)
                       USE SDTP.REF.TC;LIST:NK

                 (for Software Requirements Document)
                       USE SRD.INFO.TC;LIST:NK

                 (for Program Specifications Document)
                       USE PSD.INFO.TC;LIST:NK

                   (for Module Test Plan Document)
                     USE TSTPLAN.INFO.TC;LIST:NK

                          (for Simulation)
                     USE SIMULAT.INFO.TC;LIST:NK


 To exit ISIS enter STOP
```

Figure 10. - Shelf level

```
          use srd.info.tc;list:nk

WORK         USED FROM ASPSLIB.DOCMNT.SRD.INFO.TC
1

                                                                  1


This book  contains  a  general  document  on  the ASPS  Software
Requirements  and  specific documents for the Real-Time Executive
Module, Attitude Control Module, and the  Attitude  Determination
Module.  To  access  the   area  of  interest enter one  of   the
following commands.

                 (for ASPS Software Requirements)
                     USE REF.AGSSRD;LIST:NK

                (for Real-Time Executive Module)
                      USE RTEM.TC;LIST:NK

                  (for Attitude Control Module)
                      USE ACM.TC;LIST:NK

               (for Attitude Determination Module)
                      USE ADM.TC;LIST:NK


To exit ISIS enter STOP
```

Figure 11. - Book level

```
         use rtem.tc;list:nk

WORK         USED FROM ASPSLIB.DOCMNT.SRD.RTEM.TC
1

                                                            1


This chapter contains the Table of  Contents  for  the  Real-Time
Executive Module document.

6.0    MODULE REQUIREMENTS

6.1    Real-Time Executive Module


       The Real-Time Executive Module shall provide the  following
operating system functions for the AGS software:

       6.1.1 (I)           Program Load

       6.1.2 (II)          Program Initialization

       6.1.3 (III)         Interrupt Processing (with error
                           detection)

       6.1.4 (IV)          Program Task Scheduling

       6.1.5 (V)           I/O Data - Timing and Control

To access enter one of the following commands.

          USE I;LIST:NK

          USE II;LIST:NK

          USE III;LIST:NK

          USE IV;LIST:NK

          USE V;LIST:NK

To exit ISIS enter STOP
```

Figure 12. - Chapter level

```
        use ii;list:nk

WORK        USED FROM ASPSLIB.DOCMNT.SRD.RTEM.II
1

                                                              1
```

6.1.2 Program Initialization

The program initialization section will provide initialization of the AGS software immediately following program load. This mode shall initialize the Program Status Words (PSW's), set the storage protect keys, set up Buffered I/O, initialize the Real-Time Executive variables and zero the Real Time Clock. In addition, it will perform data and filter initialization for the designated functions listed in Table 6.1.2. When the initialization is complete, the NSSC-II will interrupt the DEA to inform it of its status and enter the "wait-state". Real-time processing will not begin until the DEA generates an interrupt to start the first real-time cycle.

Figure 13. - Page level

```
        use aspslib.testing.system.control.nsscii;list

ASPSLIB.TESTING.SYSTEM.CONTROL.NSSCII  USED AS WORK
    1.   =NSSCII,T200,CM77000.                        RM 1115  M.S. NOLAND
    2.   =USER,      .
    3.   =CHARGE,      ,LRC.
    4.   =*ICS COMPILE AND EXECUTE RUN
    5.   =ATTACH,ISTORE,IFETCH,ISISPUT,ISISGET/UN=        .
    6.   =ATTACH,PASCAL,PASCLIB/UN=LIBRARY.
    7.   =IFETCH,CKPTF.  ASPSLIB.TESTING.ICS.INTEGRT.CKPTF
    8.   =ISISGET,DATAF3.  ASPSLIB.TESTING.ICS.INTEGRT.DATAF3
    9.   =ISISGET,CONTRLF.  ASPSLIB.TESTING.ICS.INTEGRT.CONTRLF
    9.1  =ISISGET,ASPS1.  ASPSLIB.TESTING.ICS.INTEGRT.ASPS1
   10.   =REWIND,DATAF3,CONTRLF,CKPTF.
   11.   =RFL,77000.
   12.   =REDUCE,-.
   13.   =PASCAL,ASPS1.
   14.   =LGO,,,DATAF3.
   15.   =ISTORE,LGO.  ASPSLIB.TESTING.ICS.INTEGRT.LGOO
   16.   =ISTORE,CKPTF.  ASPSLIB.TESTING.ICS.INTEGRT.CKPTF
   17.   =ISISPUT,CONTRLF.  ASPSLIB.TESTING.ICS.INTEGRT.CONTRLF
   18.   =)

        USE ASPSLIB.TESTING.SYSTEM.CONTROL.NSSCII

        RUN:NK

        SEND
```

Figure 14. – Specific submit file

```
        use aspslib.testing.system.nsscii.exec;list

WORK      USED FROM ASPSLIB.TESTING.SYSTEM.NSSCII.EXEC
    1.   =ERASE WORK1; FRAME WORK1:STRING;
    2.   =ERASE S1; VAR S1:STRING;
    3.   =ERASE S; VAR S:STRING;
    4.   =ERASE SND; VAR SND:STRING;
    5.   =LOOP
    6.   =   ASK S1, ' DO YOU WANT TO EXECUTE ANOTHER JOB? <Y> OR <N> ';
    7.   =   IF (S1='Y') OR (S1='YES') THEN CLEAR RUN;
    8.   =      WORK1/USE ASPSLIB.TESTING.SYSTEM.CONTROL.NSSCII
    9.   =      WORK1/RUN:NK
   10.   =      ASK S, 'SOURCE PAGE NAME?'
   11.   =      EXITIF S='NONE';
   12.   =      XEQ CAT ('WORK1/USE ',S);
   13.   =      WORK1/RUN:NK
   14.   =      ASK SND, 'READY TO SEND?';
   15.   =      IF (SND ='YES') OR (SND='Y') THEN SEND;
   16.   =         ELSE PRINTLN ' DID NOT SEND   ',S;
   17.   =      END
   18.   =   END;
   19.   = EXITIF S1='N';
   20.   =END;
```

Figure 15. – An IPL program

```
        use aspslib.testing.system.control.nsscii;list

ASPSLIB.TESTING.SYSTEM.CONTROL.NSSCII  USED AS WORK

    1.    =NSSCII,T200,CM77000.                     RM 1115  M.S. NOLAND
    2.    =USER,          .
    3.    =CHARGE,        ,LRC.
    4.    =*ICS COMPILE AND EXECUTE RUN
    5.    =ATTACH,ISTORE,IFETCH,ISISPUT,ISISGET/UN=        .
    6.    =ATTACH,PASCAL,PASCLIB/UN=LIBRARY.
    7.    =IFETCH,CKPTF.  ASPSLIB.TESTING.ICS.INTEGRT.CKPTF
    8.    =ISISGET,DATAF3.  ASPSLIB.TESTING.ICS.INTEGRT.DATAF3
    9.    =ISISGET,CONTRLF.  ASPSLIB.TESTING.ICS.INTEGRT.CONTRLF
   10.    =REWIND,DATAF3,CONTRLF,CKPTF.
   11.    =RFL,77000.
   12.    =REDUCE,-.
   13.    =PASCAL.
   14.    =LGO,,,DATAF3.
   15.    =ISTORE,LGO.  ASPSLIB.TESTING.ICS.INTEGRT.LGOO
   16.    =ISTORE,CKPTF.  ASPSLIB.TESTING.ICS.INTEGRT.CKPTF
   17.    =ISISPUT,CONTRLF.  ASPSLIB.TESTING.ICS.INTEGRT.CONTRLF
   18.    =)
```

Figure 16. - General submit file

```
          use nsscii.exec;exec

ASPSLIB.TESTING.SYSTEM.NSSCII.EXEC  USED AS WORK
 DO YOU WANT TO EXECUTE ANOTHER JOB? <Y> OR <N> y

RUN CLEARED.
ASPSLIB.TESTING.SYSTEM.CONTROL.NSSCII  USED AS WORK1
25 ITEMS IN SPECIFIED RANGE.
SOURCE PAGE NAME? ics.integrt.asps1

ASPSLIB.TESTING.ICS.INTEGRT.ASPS1  USED AS WORK1
4258 ITEMS IN SPECIFIED RANGE.
READY TO SEND?y

"ATCYNGA" SENT TO BATCH.
 DO YOU WANT TO EXECUTE ANOTHER JOB? <Y> OR <N> n
```

Figure 17. - Execution of an IPL program

```
        use aspslib.docmnt.ssd.ref.tc;list

WORK       USED FROM ASPSLIB.DOCMNT.SSD.REF.TC
    0.001=1
    0.002=
    0.003=
    0.004=
    0.005=
    0.006=
    0.007= This book  contains  the  Table  of  Contents  for  the  Software
    0.008= Standards Document.
    0.009=
    0.01 =
    0.011=
    0.012=                         TABLE OF CONTENTS
    0.013=
    0.014=     I          Introduction                  (I)
    0.015=
    0.016=     II         Requirements Document         (II)
    0.017=
    0.018=     III        Program Specification         (III)
    0.019=
    0.02 =     IV         Detail Internal Specification (IV)
    0.021=
    0.022=     V          Debug/Test                    (V)
    0.023=
    0.024=
```

Figure 18. – A page of documentation in ASPSLIB

1. Report No. NASA TM-81924	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle An Evaluation of the Interactive Software Invocation System (ISIS) for Software Development Applications		5. Report Date January 1981
		6. Performing Organization Code 506-61-43-05
7. Author(s) Marie S. Noland		8. Performing Organization Report No.
		10. Work Unit No.
9. Performing Organization Name and Address NASA Langley Research Center Hampton, Virginia 23665		11. Contract or Grant No.
		13. Type of Report and Period Covered Technical Memorandum
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, DC 20456		14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

The Multipurpose User-Oriented Software Technology (MUST) project was established to reduce the cost of flight software and effectively utilize digital systems to support flight research. One of the tools designed and built under this project was the Interactive Software Invocation System (ISIS). ISIS was developed primarily to handle large amounts of information required for the support of flight projects. The Annular Suspension and Pointing System (ASPS) flight project, which contains a vast amount of information, was integrated into ISIS. This resulted in an integrated system capable of performing the verification and validation requirements of the ASPS flight code. This system consists of language processors, automated documentation tools, utility routines, a flowcharter, and a test and simulation system. This ordered collection of information is controlled through the interactive user interface capability provided by ISIS.

17. Key Words (Suggested by Author(s)) APSP, ISIS, MUST, Flight Software, Hierarchical Library, IPL, Text Editor, File Manager, Tool Invoker, NOS	18. Distribution Statement Unclassified – Unlimited Subject Category 61

19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 29	22. Price* A03

N-305

*For sale by the National Technical Information Service, Springfield, Virginia 22161