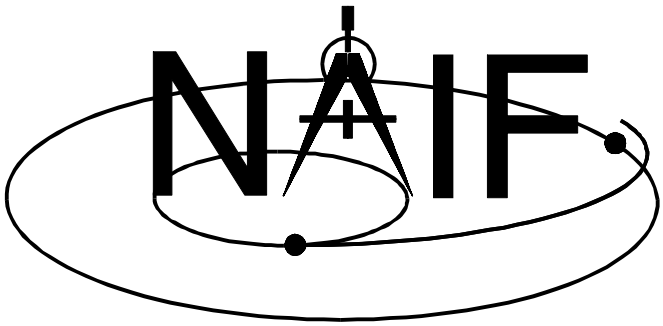
Introduction to SPICE



NAIF

January 1997

Welcome...

Welcome to the SPICE system. This system allows you to easily combine the most accurate space geometry and event data with your mission analysis, observation planning or science data processing software.

SPICE has gained wide acceptance for organizing, distributing and accessing space geometry and event data. As you move on from your current investigations to other planetary science studies you will likely find SPICE available to assist you in your new endeavors.

The SPICE system is available for the following computers:

- VAX/VMS systems
- DEC ALPHA OpenVMS and OFS/1 workstations
- SUN workstations
- HP workstations
- SGI workstations
- IBM-PCs and compatibles
- Apple Macintosh computers
- NeXT workstations

Since the source code of the software portion of the SPICE system is written in ANSI Standard FORTRAN 77, the system can be easily ported to other environments that support this language.

The SPICE system has many components. This overview is designed to help guide you through the various components and to point to other, more extensive, documentation.

Quality and Stability

We here at NAIF are committed to providing a robust, flexible, and understandable product that suits the needs of SPICE users. The software that makes up the SPICE toolkit undergoes careful review and testing. The documentation for the SPICE software is extensive and accurate. And, we listen to input from users like you.

In addition to listening to users comments, we use the product ourselves. As a result, we often can anticipate needs that you will encounter as you learn and use the SPICE system.

The SPICE system is a work in progress. However, NAIF is equally committed to providing SPICE users a ***stable*** product. We understand the frustrations of learning a system only to have it change in unanticipated ways. For this reason we ***never*** change the intended functionality of, or the interface to released software and data products. If you build a program using today's SPICE system, it will work the same way when you re-build it with tomorrow's SPICE system.

Installation

To make use of the SPICE system you need to install it on your computer. This means that you must execute a script or command procedure (provided with your SPICE

package) that creates the SPICE directory system and populates this directory structure with the various files that make up the SPICE system. Some of these files will be created as a result of the installation. These files are the object libraries and executable programs that are compiled and linked using the FORTRAN source code that comes with the SPICE toolkit. The time required for installation varies with the speed of your computer and its FORTRAN compiler. On most systems, installation can be done in under an hour.

After installation all SPICE system documents mentioned in this overview can be found under the ***doc*** subdirectory.

Contents of the Toolkit

Once you've installed the SPICE system on your computer, you can find out what you have by consulting the document ***dscriptn.txt***. This document describes the directory structure created by the installation procedure and the contents of the various directories.

Using SPICE

The only way to make use of the SPICE system is to write programs that call subroutines in the SPICE toolkit. To do this you will need at a minimum:

1) some form of text editor for creating programs that call subroutines in SPICELIB — the SPICE FORTRAN subroutine library. Its object module, called ***spicelib***, is located under the ***lib*** subdirectory. The source code files for all SPICELIB subroutines are located under the ***src/spicelib*** subdirectory ;
2) a compiler for your computer that is capable of calling FORTRAN subroutines; and
3) a linker for building executable programs.

You will need to be familiar with using your computer's editor, compiler, and linker. Familiarity with a debugger can also be helpful as you develop your application program. Finally, you will need to know how to execute a program on your computer.

In future releases of the SPICE system, NAIF plans to provide documentation that describes how to do this in the various computing environments for which we actively support the SPICE system. But for now, if any of these topics (editing, compiling or linking) are not clear to you, try examining the routines used by the installation procedure to build the various SPICE applications. For example, examine the procedure ***mkprodct*** (or ***mkspacit*** on some computer platforms) that is located in the source code directory for SPACIT (see ***dscriptn.txt*** to locate this directory). If this example isn't sufficient to clear up any questions, we recommend that you consult a colleague or your system administrator to find out the details of program development on your computer.

Bilingual Programming

If you "speak" FORTRAN, the easiest way to create programs that call routines in SPICELIB is to write your programs in FORTRAN. However, you may wish to write programs in another programming language (for example C, C++ or Pascal). This can be a bit tricky. Matrices and strings may be organized differently in your language of

choice from the way they are organized in FORTRAN. Nevertheless, this may be the shortest path to your ultimate goal.

If you choose this path you will need to find out how your language of choice supports calls to FORTRAN object modules. Your best source of information will be a colleague who has already mastered this art. There may be a programmer's reference manual that describes cross–language syntax. In addition, you may be able to receive support from the company that supplies the language you choose for writing your software.

Files

The basic unit for organizing information on your computer is the file. The SPICE system is composed of several different types of files. The two basic types are text files and binary files.

Text files are interpreted as simply a string of ASCII characters with special markers to indicate how line breaks and spacing should be arranged.

Binary files on the other hand have more specialized organization. They typically contain numeric and textual information. The numeric information is stored in the binary format used by your computer instead of a form that is easily read by people.

Text files can be examined or modified in a text editor. On the other hand, binary files can typically be examined or modified only through use of specialized software that understands the particular organization of the binary file.

Because, text files have a nearly universal format, they can be transported easily from one machine to another. There are several readily available programs available to assist with this task. For this reason, the SPICE system is almost always distributed as a collection of text files.

The text files that make up the delivered product fall into several categories: documentation, source code, make or link procedures, and some sample data.

Documentation that is of a generic nature is collected under ***doc*** subdirectory. These files include user's guides for the various applications, index information, and discussion of related collections of routines that make up SPICELIB.

Documentation on a subroutine by subroutine basis is located in the source code for the routine. The subroutine documentation is in the form of an extensive "header" that appears at the beginning of the source file. All SPICELIB source files are located under the ***src/spicelib*** subdirectory.

All source files are compiled to produce object files. Most of the object files are gathered into an object library called ***spicelib*** located under the ***lib*** subdirectory. You will use this object library when creating your own programs. Other object files are linked with SPICELIB to create the application tools that are part of the SPICE system.

Sample data files make up the remaining files. Some of these (LSK and PCK) can be used as you receive them. Others must be converted from transfer to binary format before you can use them with toolkit software; these are the ephemeris (SPK), attitude (CK) and spacecraft events (EK) information files. The conversion programs used for converting these files are called ***spacit***, ***tobin*** and ***toxfr***. They are provided with the

SPICE system and their executable modules are located under the ***exe*** subdirectory. They are discussed briefly in the TOOLS section below and are documented in the ***spacit.ug*** and ***convrt.ug*** User's Guide documents that come with SPICE.

Kernels

The data files (both binary and text) that are used by the SPICE system are called ***kernels***. The binary kernels are the SP-kernels (SPK), a few PC-kernels (PCK), C-kernels (CK) and some E-kernels (EK). The binary kernels are often delivered in the SPICE "transfer format" which is portable between all computers. You will need to convert such files to your computer's binary format using either ***tobin*** or ***spacit*** utility program delivered with the toolkit. There are also a handful of text kernels: Spacecraft Clock kernels (SCLK), Leapseconds kernel (LSK), most Physical Constants kernels (PCK), Instrument parameter kernels (IK) and some spacecraft Events kernels (EK). These can be used as received — no conversion is needed.

Both types of kernels (text and binary) must be read to be useful. The SPICE system provides software for reading (and in some cases writing) these kernels. You can use these kernels without ever concerning yourself with the details of their format.

To make the information in kernels available to your software you will need to ***load*** them. This lets the SPICE system know where to find these data. Loading text kernels is discussed in the document ***kernels.req***. Loading SP-kernels is discussed in the document ***spk.req***. Loading C-kernels is discussed in the document ***ck.req***. Loading PC-kernels is discussed in the document ***pck.req***. Moreover examples of loading kernels are provided in the cookbook programs that are provided with SPICE (see the section on Example Programs below). Kernels of all types need to be loaded only once in your program — don't put the load statement in a loop.

Creating or Modifying Kernels

Most SPICE user's will not need to create or modify kernels. Moreover, of the users who need to create or modify kernels, most will only need to deal with one of the various text kernels. Because these are text files they can be created and modified with any text editor. Although the format of these files is fairly easy to deduce, we recommend that you consult the document ***kernel.req*** before you attempt to edit a text kernel for the first time.

Some of SPICE users will want to create binary kernels. This task requires a deeper understanding of the SPICE kernel format and kernel readers. If you need to create SPICE binary kernels we recommend that you read all of the material in the one of ***spk.req***, ***ck.req***, ***pck.req*** or ***ek.req*** that is pertinent to your task. In addition, if you are producing these data products in an official project capacity we encourage you to contact NAIF to discuss any issues that might be relevant to your task of producing binary kernels.

Obtaining Kernels

Since you will probably not be producing your own kernels, you will need to obtain them from someplace. Usually, you will receive them from a project database or project data management team. Once a mission is complete the kernels should be

available from a national data archive center. The NAIF node of the Planetary Data System is one such example.

Programming Examples

We often learn new ideas best by seeing examples of their use. For this reason SPICE comes with a collection of example programs called the ***SPICE Cookbook***, located under the ***cookbook*** subdirectory. These sample programs show in a page or so how to load various kernel files, how to call the kernel readers and how to perform common geometric and time computations. Most user's find these to be very valuable in getting started with the SPICE system. We recommend that you look at the ***Cookbook*** early to help you get an idea of how easily you can begin programming using SPICE.

Tools

A handful of utility programs are provided to assist you in your use of SPICE. These programs are:

SPACIT Used to convert binary format kernels to a "transfer" format (and vice-versa) for porting between different environments. Also used for summarizing the contents of binary kernels. (See the ***SPACIT User's Guide.***)

INSPEKT Used for examining the contents of binary EK kernels. (See the ***INSPEKT User's Guide***. This is available only as a printed document or PostScript file.)

TOBIN Used to convert "transfer" format kernels to a binary format using a simple command line interface. (See the ***CONVERT User's Guide.***)

TOXFR Used to convert binary format kernels to a transfer format using a simple command line interface. (See the ***CONVERT User's Guide.***)

COMMNT Used to provide internal documentation in binary kernels. (See the ***COMMNT User's Guide***.)

The executable modules of these programs are located under the ***exe*** subdirectory.

Concepts and Terminology

The SPICE system is very much concerned with concepts of geometry and time. To talk about geometry and time the planetary science community uses several different time and coordinate systems. Within the context of a single planetary mission, one does not often need to be concerned with the details of these systems. Project management simply dictates a common system to be used for all software and data products. However, the scope of the SPICE system spans many planetary missions and terrestrial based investigations. As a result, you will need to pay more careful attention to the details of the various measurement systems.

NAIF has provided mechanisms for converting between different time systems such as UTC, JED, TDB and Spacecraft Clock. SPICE also provides conversions between

various inertial reference frames such as J2000, B1950, EME50, and non-inertial frames such as body-fixed frames.

For more detailed information consult:

naif_ids.req	for a discussion of the coordinate frames supported within SPICE.
rotations.req	for a discussion of the mathematical properties of rotations and the software for manipulating them.
sclk.req	for information about time as measured by the spacecraft clock.
time.req	for detailed information about conversion between various time systems

Better Programming

The SPICE system is much more than kernels and kernel readers. Over half the toolkit is devoted to routines that make the process of building software easier. Those familiar with the standard C library will find a find a parallel between some SPICE FORTRAN routines and standard C functions.

Using the toolkit for more than simply getting data means that you can often produce more robust software in less time than had you not used the toolkit routines. There are some obvious reasons for this. You don't have to write the software that duplicates toolkit functions. Toolkit software has already undergone careful testing. The action of toolkit software is carefully specified. In addition, you benefit from a centralized error handling facility that is integrated with toolkit software.

To find out more about the various features that are part of the toolkit, consult the following documents that can be found under the ***doc*** subdirectory:

cells.req	Discusses a data structure based on arrays that carries declaration information about the array.
ck.req	Discusses the SPICE system for providing attitude information, usually for a spacecraft.
daf.req	Discusses the file architecture that is the basis for both SPK and CK file formats. It can be used anywhere you need to store arrays of floating point numbers in a file.
ellipses.req	Discusses a data structure for ellipses and routines for manipulating them.
error.req	Discusses the central error and exception handling facility that is built into the SPICE system.
naif_ids.req	Discusses the NAIF numbering system for solar system objects, reference frames, spacecraft and spacecraft structures, and instruments.
pck.req	Discusses models used for rotations of solar system bodies, their shapes and how to put this information into a text kernel.
planes.req	Discusses a data structure for representing geometric planes and software for manipulating them.

rotations.req	Discusses the representation and mathematics of rotations and software for manipulating them.
scanning.req	Discusses routines in SPICELIB for manipulating character strings.
sclk.req	Discusses spacecraft clocks, their representations, and the spacecraft clock kernel.
sets.req	Discusses the SPICE data structure for representing sets of numbers or character strings and software for performing basic set operations.
spc.req	Discusses tools for tagging SPK and CK files with internal comments.
spk.req	Discusses the NAIF ephemeris system.
symbols.req	Discusses a data structure that is used to associate names with data and software for manipulating these structures.
time.req	Discusses the use of time in SPICE and the routines available for converting between different representations.
windows.req	Discusses a data structure used to represent subsets of the real line composed of disjoint closed intervals and the software available for performing set operations on them. These are often used to represent intervals of time.

Finding the right routine

The real programming power of SPICELIB comes from the extensive set of built in capabilities that you don't have to re-invent. However, SPICELIB contains over 900 routines. Reading all of the headers for these routines is not an acceptable way to locate the routine that does the job you want to do. If you can classify the function you need as belonging to one of the categories listed in the previous section, then you can consult one of the listed documents. However, there are many routines in SPICELIB that don't fit into one of the categories listed above. (For example there are over 40 routines for performing vector-matrix arithmetic.) With this indexing problem in mind, we have provided additional documents to help you quickly locate the right routine for the task at hand.

The document ***spicelib.idx*** provides a permuted index to all of the routines in SPICELIB. Like the index to a reference work, this helps you find the right routine for a specific task. Moreover, the index is permuted. Thus, the word you think of as an index to a routine has a much greater chance of pointing to a relevant routine. For example you will find:

Cross product of two vectors
Product of two vectors, cross
Vectors, cross product of two

all listed in the permuted index. Each of these point to the SPICELIB subroutine VCRSS that computes a cross product.

What to do when you have a problem

With any new product people sometime encounter problems. If you run into a problems here's some steps that you can take to try to resolve them.

If you think the problem is related to a SPICE routine, look at the documents related to that routine (these are listed in the required reading section of the routine's header). These documents may offer information to help you in diagnosing and correcting the problem.

If the problem seems to be environmental (for example you can't link a program, can't find a file, don't have sufficient privileges to perform some action, etc.) contact your system manager. (S)he may be able to assist you. If the problem is truly an environmental one, NAIF will probably not be able to provide much more than sympathy.

If you are not using SPICE directly, but using a product developed by someone else, contact that person to see if they can help out.

If you don't seem to have the correct kernels or they don't seem to be acting as you expected, make sure you are using a text file where text is called for and a binary file where binary files are called for.

If your program is working but your numbers don't agree with those of a colleague, make sure you are using the same kernel files.

If you don't have the latest version of a kernel associated with a flight project (or want to find out if you do) contact the project data administrator.

If you can't seem to fit the problem into a category above, or just aren't getting anywhere, send us (NAIF) e-mail or call us on the phone. We'll do what we can within our resources to help out.

Let us know what you think

We are interested in your comments about the SPICE system. If you like what you've received so far, let us know. If there's something wrong tell us that too. If you have ideas for improvements or enhancements, we'd like to hear about them. With the support and suggestions of users such as yourself, the SPICE system will be a valuable resource for the space science community into the next century.