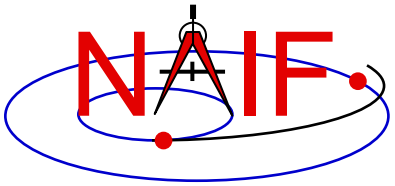
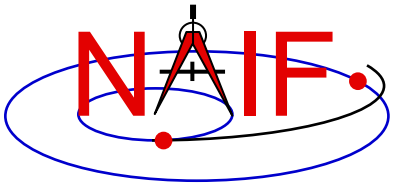


Navigation and Ancillary Information Facility

Introduction to Kernels

January 2017



Agenda

- Overview
- Kernel architecture
- Producing kernels
- Using kernels

What is a SPICE “Kernel”

“Kernel” means file

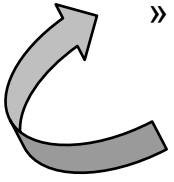
“Kernel” means a file containing ancillary data

“Kernel” means a file containing "low level" ancillary data that may be used, along with other data and SPICE Toolkit software, to determine higher level observation geometry parameters of use to scientists and engineers in planning and carrying out space missions, and analyzing data returned from missions.

SPICE Kernels Family

Navigation and Ancillary Information Facility

- SPK
  - Spacecraft and Planet Ephemeris
- PCK
  - Planetary Constants, for natural bodies
    - » Orientation
    - » Size and shape
- IK
  - Instrument
- CK
  - Pointing ("C-matrix")
- EK
  - Events, up to three distinct components
    - » ESP: science plan
    - » ESQ: sequence
    - » ENB: experimenter's notebook



EK is rarely used

- FK
  - Reference frame specifications
- SCLK
  - Spacecraft clock correlation data
- LSK
  - Leapseconds
- MK
  - Meta-Kernel (a.k.a. "FURNSH kernel")
  - Mechanism for aggregating and easily loading a collection of kernel files
- DSK
  - Digital shape kernel
    - » Tesselated plate model
    - » Digital elevation model
- DBK
  - Database mechanism
    - » Primarily used to support the ESQ

Text and Binary Kernels

Navigation and Ancillary Information Facility

SPICE text kernels are:

- text PCK (the most common type of PCK)
- IK
- FK
- LSK
- SCLK
- MK

SPICE binary kernels are:

- SPK
- binary PCK (has been used only for Earth, moon and Eros)
- CK
- DSK
- ESQ (part of the E-kernel)
- DBK (database kernel)

Rarely used

SPICE Kernel Forms

Navigation and Ancillary Information Facility

- **Binary form: SPK, binary PCK, CK, EK/ESQ[1], DSK**
  - A file mostly containing data encoded in binary form
  - Binary kernels are not human-readable; they require the use of Toolkit software to examine the data contents
- **Text form: text PCK, IK, FK, LSK, SCLK, MK**
  - Plain text files containing only printing characters (ASCII values 32-126), i.e. human-readable text.
- **"Transfer" form of a binary kernel**
  - This is an ASCII representation of a binary kernel
  - Was used for porting the file between computers with incompatible binary representations (e.g. PC and UNIX)
  - Use of the transfer format is no longer needed for porting due to the run-time translation capability added to SPICE long ago
    - » But it is one way to convert a non-native binary kernel into native format, needed for modifying the kernel or improving read efficiency

[1] The ESP and ENB components of the EK might be binary, or text, or html, depending on specific implementation.

Kernel Architecture

- Text kernels
- Binary kernels
- Comments in kernels

Text Kernel Contents

- A text kernel is a plain text file of ASCII data
- It contains assignments of the form:

```
variable_name = value(s)
```

- A text kernel should also contain descriptive comments that describe the assignments
  - Comments are sometimes referred to as “meta-data”
    - » Don’t confuse this usage with the “meta-kernel” described later in this tutorial

Example Text Kernel

Navigation and Ancillary Information Facility

```
KPL/<kernel type>
 \begindata

   NAME             = 'Sample text value'
   NaMe             = 'Keywords are case sensitive'

   NUMBERS          = ( 10.123, +151.241, -1D14 )
   NUMBERS         += ( 1.0,    1,         -10   )
   NUMBERS         += ( 1.542E-12, 1.123125412  )

   START            = @2011-JAN-1

   \begintext

      < some comments about the data >

   \begindata

      < more data in keyword = value syntax >

   \begintext
         < etc., etc. >
```

A data block

A “comments” block

Another data block

Another “comments” block

- **The next several pages describe what you see above**
- **See the “Kernel Required Reading” document for details**

Text Kernel Formatting

- `KPL/<kernel type>`
  - Its use is optional, but is highly recommended
  - Must appear on the first line, starting in column 1
  - Tells SPICE software what kind of kernel it is
  - Examples of kernel type are FK, IK, PCK, SCLK

- `\begindata` and `\begintext`
  - Markers, on lines by themselves, which set off the beginning of data and the beginning of meta-data blocks respectively
  - They need not begin in column 1

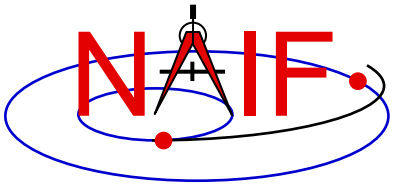
- `<LF>` for Unix/Linux/Mac or `<CR><LF>` for Windows
  - End of line marker (usually not visible when displaying a text kernel)
  - Must be present on EVERY line in the text kernel

- Max line length, including any white space is 132 characters

Text Kernel Operators

- An assignment using the "=" operator associates one or more values with a variable name.
- An assignment using the "+=" operator associates additional values with an existing variable name.
- An assignment using the "@" symbol associates a calendar date with a variable name.
  - The string will be parsed and converted to an internal double precision representation of that epoch as seconds past the J2000 epoch
    - » There is no time system implied
    - » This conversion does not need a leap seconds kernel

Using Kernels



Loading Kernels - 1

- To make kernels available to a program you “load” them

- When you load a text kernel:
  - the file is opened
  - the kernel contents are read into memory
    - » variable names and associated values are stored in a data structure called the “kernel pool”
  - the file is closed

- When you load a binary kernel:
  - the file is opened
  - for SPK, CK, and binary PCK files, no data are read until a read request is made by Toolkit software
  - for ESQ files, the schema description is read, checked, and stored in memory at load time, but no data are read until a query/fetch is made
  - for all practical purposes the binary file remains open unless specifically unloaded by you

Loading Kernels - 2

- **Use the FURNSH routine to load all kernels–text and binary**
  - `CALL FURNSH ('name.ext' )` (Fortran)
  - `furnsh_c ("name.ext" );` (C)
  - `cspice_furnsh, 'name.ext'` (IDL)
  - `cspice_furnsh ('name.ext' )` (MATLAB)

- **Best practice: don't hard code filenames–list these in a "meta-kernel" and load the meta-kernel using FURNSH**
  - `CALL FURNSH ('meta-kernel_name')` (Fortran example)
  - **See the next page for more information on meta-kernels**

- **Caution: "Transfer format" versions of binary kernels can not be loaded; they must first be converted to binary with the Toolkit utility program *tobin* or *spacit***

Meta-Kernels

What is a “Meta-Kernel”

- A meta-kernel is a file that lists names (and locations) of a collection of SPICE kernels that are to be used together in a SPICE-based application
  - You can simply load the meta-kernel, causing all of the kernels listed in it to be loaded
- Using a meta-kernel makes it easy to manage which SPICE files are loaded into your program
- A meta-kernel is implemented using the SPICE text kernel standards
  - Refer to the Kernel Required Reading technical reference for details
- The terms “meta-kernel” and “FURNSH kernel” are used synonymously

Sample Meta-Kernel


```
KPL/MK
\begindata
    KERNELS_TO_LOAD = (
              '/home/mydir/kernels/lowest_priority.bsp',
              '/home/mydir/kernels/next_priority.bsp',
              '/home/mydir/kernels/highest_priority.bsp',
              '/home/mydir/kernels/leapseconds.tls',
              '/home/mydir/kernels/sclk.tsc',
              '/home/mydir/kernels/c-kernel.bc',
              '/home/mydir/kernels+',
              '/custom/kernel_data/p_constants.tpc',
                       )
```

The commas are optional

Sample Meta-Kernel


```
KPL/MK
\begindata
    KERNELS_TO_LOAD = (
                '/home/mydir/kernels/lowest_priority.bsp',
                '/home/mydir/kernels/next_priority.bsp',
                '/home/mydir/kernels/highest_priority.bsp',
                '/home/mydir/kernels/leapseconds.tls',
                '/home/mydir/kernels/sclk.tsc',
                '/home/mydir/kernels/c-kernel.bc',
                '/home/mydir/kernels+',
                '/custom/kernel_data/p_constants.tpc',
                        )
```

The commas are optional

• The last file listed in this example (p_constants.tpc) demonstrates how to use the continuation character, '+', to work around the 80 character limitation imposed on string sizes by the text kernel standards.

• See the next two pages for some important OS-specific details!

Unix/Mac
Sample Meta-Kernel

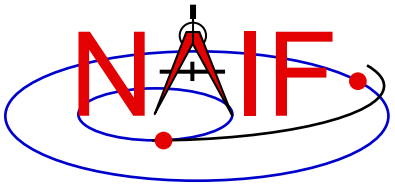
• This meta-kernel uses the PATH_VALUES and PATH_SYMBOLS keywords to specify the directory where the kernels are located.

```
KPL/MK
\begindata
   PATH_VALUES     = ( '/home/mydir/kernels' )
   PATH_SYMBOLS    = ( 'KERNELS'             )
   KERNELS_TO_LOAD = (
          '$KERNELS/lowest_priority.bsp',
          '$KERNELS/next_priority.bsp',
          '$KERNELS/highest_priority.bsp',
          '$KERNELS/leapseconds.tls',
          '$KERNELS/sclk.tsc',
          '$KERNELS/c-kernel.bc',
          '$KERNELS/custom/kernel_data/p_constants.tpc'
                          )
```

UNIX/MAC style path notation (forward slashes)

• Although the OS environment variable notation $<name> is used to refer to the symbols specified using the PATH_VALUES and PATH_SYMBOLS keywords, these symbols are NOT operating system environment variables and are set and used for substitution by SPICE only in the context of this particular meta-kernel.

• The '+' continuation character described on the previous page may be used to handle path strings that exceed 80 characters.



Windows
Sample Meta-Kernel

Navigation and Ancillary Information Facility

• This meta-kernel uses the PATH_VALUES and PATH_SYMBOLS keywords to specify the directory where the kernels are located.

```
KPL/MK
\begindata
   PATH_VALUES     = ( '\home\mydir\kernels' )
   PATH_SYMBOLS    = ( 'KERNELS'             )
   KERNELS_TO_LOAD = (
          '$KERNELS\lowest_priority.bsp',
          '$KERNELS\next_priority.bsp',
          '$KERNELS\highest_priority.bsp',
          '$KERNELS\leapseconds.tls',
          '$KERNELS\sclk.tsc',
          '$KERNELS\c-kernel.bc',
          '$KERNELS\custom\kernel_data\p_constants.tpc'
                          )
```

Windows style path notation (backwards slashes)

• Although the OS environment variable notation $<name> is used to refer to the symbols specified using the PATH_VALUES and PATH_SYMBOLS keywords, these symbols are NOT operating system environment variables and are set and used for substitution by SPICE only in the context of this particular meta-kernel.

• The '+' continuation character described previously may be used to handle path strings that exceed 80 characters.

Kernel Precedence Rule

- **The order in which SPICE kernels are loaded at run-time determines their priority when requests for data are made**
  - **For binary kernels, data from a higher priority file will be used in the case when two or more files contain data overlapping in time for a given object.**
    - **» For SPKs, CKs and binary PCKs the file loaded last takes precedence (has higher priority).**
    - **» Priority doesn't apply to ESQ files – all data from all loaded files are available.**
  - **If two (or more) text kernels assign value(s) to a single keyword using the "=" operator, the data value(s) associated with the last loaded occurrence of the keyword are used–all earlier values have been replaced with the last loaded value(s).**
  - **Orientation data from a binary PCK always supersedes orientation data (for the same object) obtained from a text PCK, no matter the order in which the kernels are loaded.**

Unloading Kernels

Navigation and Ancillary Information Facility

- The unloading of a kernel is infrequently needed for FORTRAN or CSPICE applications but is essential for Icy and Mice scripts
  - Because of the way IDL and MATLAB interact with external shared object libraries any kernels loaded during an IDL or MATLAB session will stay loaded until the end of the session unless they are specifically unloaded
- The routines KCLEAR and UNLOAD may be used to unload kernels containing data you wish to be no longer available to your program.
  - KCLEAR unloads all kernels and clears the kernel pool
  - UNLOAD unloads specified kernels
  - KCLEAR and UNLOAD are only capable of unloading kernels that have been loaded with the routine FURNSH. They will not unload any files that have been loaded with older load routines such as SPKLEF (those used prior to availability of FURNSH).
- Caution: unloading text kernels with UNLOAD will also remove any kernel pool data provided through the kernel pool APIs (PCPOOL, PDPOOL, PIPOOL)