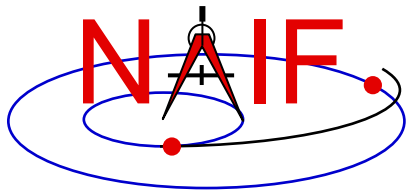


Navigation and Ancillary Information Facility

# Making an SPK File

January 2018

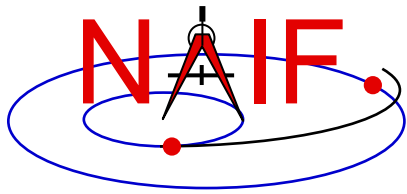


# Table of Contents

---

## Navigation and Ancillary Information Facility

- **Purpose**
- **Scope**
- **Assumptions about user's knowledge**
- **SPK overview**
- **Summary of SPK architecture**
- **Discussion applicable to all production methods**
  - Recommended SPK types
- **Selecting the polynomial degree (for polynomial SPK types)**
- **SPK production methods**
  - Using the “Make SPK” (MKSPK) program
  - Using SPICELIB, CSPICE or IDL writer modules (subroutines)
- **Finishing up, applicable to all methods**
  - Adding comments
  - Validation
  - Merging
- **Special requirements for making SPKs to be used in DSN/SPS software for view period generation, scheduling, metric predicts generation, and related functions.**
  - Applies only to those entities not making JPL NAV-style “p-files”
- **Issues affecting performance (reading efficiency) and usability**

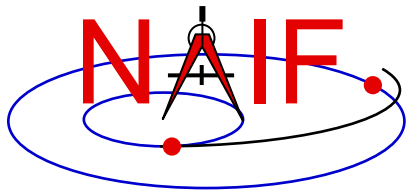


# Purpose

---

Navigation and Ancillary Information Facility

- **This tutorial provides guidance for writing an SPK file using software provided by NAIF:**
  - using the MKSPK application program
  - or
  - using SPK writer modules from SPICELIB (FORTRAN), CSPICE (C-language), Icy (IDL) or Mice (MATLAB) toolkit
    - » Only partial implementation in Icy
    - » Only Type 8 writer implemented in Mice

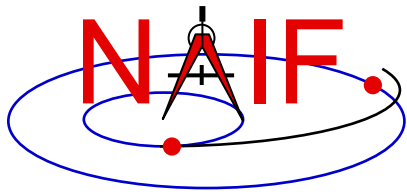


# Scope

Navigation and Ancillary Information Facility

- **This tutorial addresses production of SPKs:**
  - for general purposes
  - for use with NASA's Deep Space Network (the "SPS")
- **This tutorial does not address production of SPKs:**
  - by JPL navigation teams using the NIOSPK utility
    - » Those NAV teams may simply learn how to use the NIOSPK program and any useful SPK-related utilities.
  - by ESA/ESAC for its missions that use SPICE
    - » Those use custom "MEX2KER" software that was provided by NAIF
  - having TCB time tags, in support of certain IAU needs
    - » These are SPK Types 101, 102, etc.
  - use of the OEM2SPK utility, used to process CCSDS\* Orbit Ephemeris Message files
    - » See the OEM2SPK User's Guide for this information

\*CCSDS = Consultative Committee for Space Data Systems

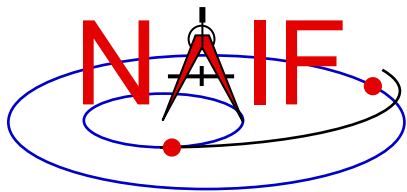


# Background Assumptions

---

Navigation and Ancillary Information Facility

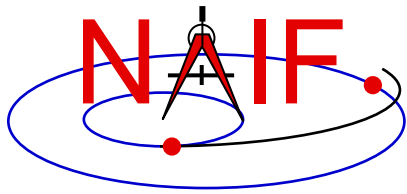
- **It is assumed the reader has some familiarity with the SPICE system, and with basic ideas of orbital mechanics.**
  - The SPICE Overview tutorial is available at:  
[ftp://naif.jpl.nasa.gov/pub/naif/toolkit\\_docs/Tutorials/](ftp://naif.jpl.nasa.gov/pub/naif/toolkit_docs/Tutorials/) (named 03\_spice\_overview)
- **It is assumed the reader has read the “SPK Tutorial” that characterizes much of the SPK subsystem, with emphasis on reading SPK files.**
  - The SPK “reading” tutorial is available at:  
[ftp://naif.jpl.nasa.gov/pub/naif/toolkit\\_docs/Tutorials/](ftp://naif.jpl.nasa.gov/pub/naif/toolkit_docs/Tutorials/) (named 19\_spk)
- **It is assumed the reader has available the SPK reference document entitled “SPK Required Reading,” supplied with each copy of the SPICE Toolkit (.../doc/spk.req)**
  - SPK Required Reading is also available at:  
<http://naif.jpl.nasa.gov/naif/documentation.html>



# SPK References

Navigation and Ancillary Information Facility

- **References for SPK production**
  - “Making an SPK” tutorial (this document)
  - “SPK (Ephemeris System)” tutorial (focused on reading an SPK)
  - “SPK Required Reading” technical reference ([spk.req](#))
  - “MKSPK Users Guide” ([mkspk.ug](#))
  - The source code “headers” provided as part of the SPK writer modules (subroutines)
  - “SPKMERGE User’s Guide” ([spkmerge.ug](#))
  - “SPY User’s Guide” ([spy.ug](#))

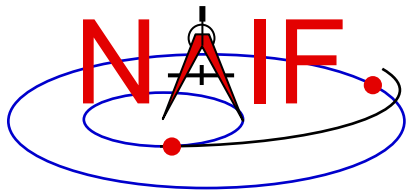


# Brief Overview - 1

---

Navigation and Ancillary Information Facility

- **You need to understand the physics of your data and how that relates to SPK type. For instance:**
  - Type 5 works best for an orbit well approximated by a sequence of one or more conic orbits.
  - Types 9 and 13 fit data regardless of the expected physics.
    - **Caution:** a good fit in the mathematical realm may not respect the physics of the trajectory. For example, fitting polynomials to an excessively sparse set of states for a planetary orbiter could result in an interpolated path that intersects the planet.

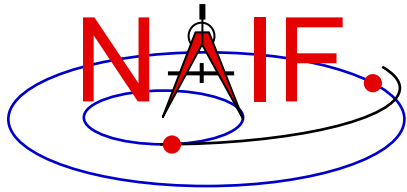


# Brief Overview - 2

Navigation and Ancillary Information Facility

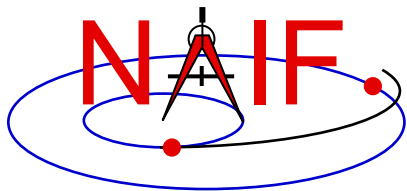
- **Ordinarily, use the NAIF MKSPK application to create SPKs from Cartesian state data or conic elements.**
  - **SPK types 5, 9, 10, and 13 are the most commonly used.**
    - » **Type 5, yields compact SPK files when the trajectory is well approximated by piecewise two-body motion. Type 5 may be the best choice for planetary or solar orbiters when available state data are sparse.**
    - » **Type 9, a good, general choice**
    - » **Type 13, when you have very accurate velocity data**
    - » **Type 10 applies ONLY to Two Line Element Sets (TLEs).**
- **Alternatively, use the Toolkit's SPK writing subroutines in your own production program.**
- **Caution:** an SPK made for use by the NASA DSN has additional requirements, discussed later on.





Navigation and Ancillary Information Facility

# Summary of SPK Architecture



# SPK File Structure: The User's View

Navigation and Ancillary Information Facility

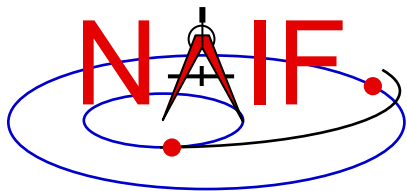
## Logical Organization of an SPK File



Always present

Possibly present

- sometimes by your choice
- sometimes required



# SPK File Structure - 1

Navigation and Ancillary Information Facility

## A minimal SPK file, containing only one segment

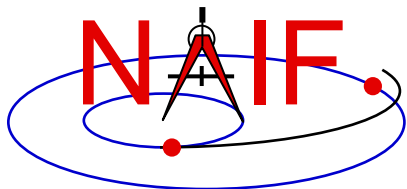
Records are fixed-length: 1024 bytes

ID WORD	ND	NI	IFNAME	FWD	BWD	FREE	BFF	0 PAD	FTP	0 PAD
Comment area text									U*	
N/P/C	D1	U								
I1	U									
Segment 1										

- ▶ **File record:** One record.
- ▶ **Comment area:** Present only if used. If used, one or more records.
- ▶ **Descriptor record:** Contains 1 to 25 segment descriptors. One record.
- ▶ **Segment ID record:** Contains 1 to 25 segment IDs. One record.
- ▶ **Data segment:** One or more records.

**ID WORD:** Indicates file architecture and type  
**ND, NI:** Number of d.p. and integer descriptor components  
**IFNAME:** Internal file name  
**FWD, BWD:** Forward and backward linked list pointers  
**FREE:** First free DAF address  
**BFF:** Binary file format ID  
**FTP:** FTP corruption test string

**N/P/C:** Next, previous record pointers and descriptor count  
**Dn:** Descriptor for segment n  
**In:** Segment ID for segment n  
  
**U:** Unused space  
**U\*:** Possibly unused space

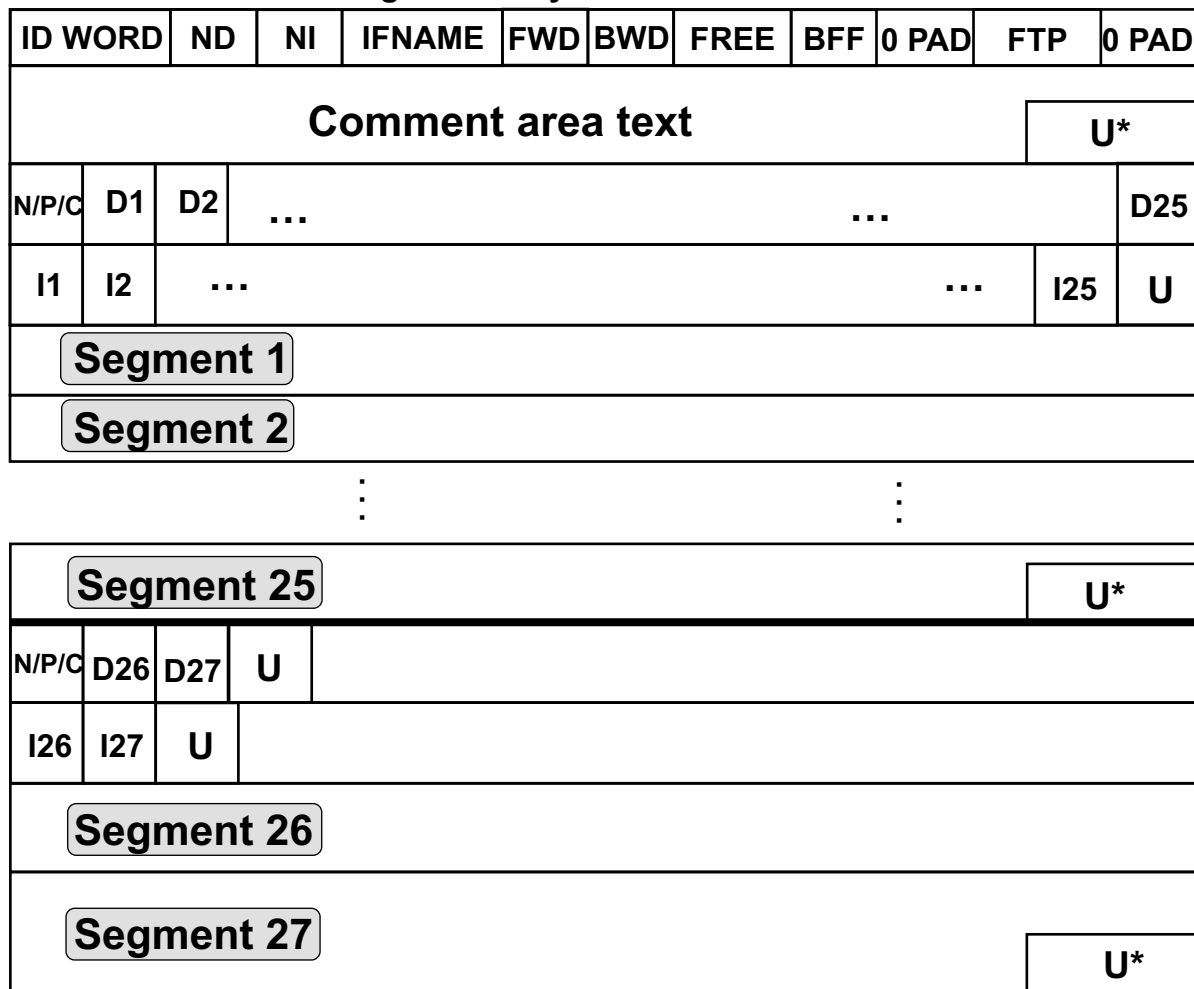


# SPK File Structure - 2

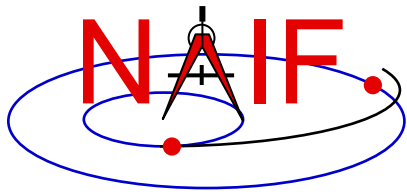
Navigation and Ancillary Information Facility

An SPK file containing multiple segments, in this example, 27

Records are fixed-length: 1024 bytes



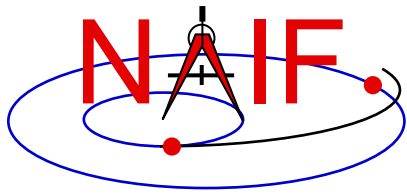
- ▶ **File record:** One record
- ▶ **Comment area:** Present only if used.
- ▶ **Descriptor record:** Contains 1 to 25 segment descriptors. One record.
- ▶ **Segment ID record:** Contains 1 to 25 segment IDs. One or more records.
- ▶ **Data segments:** One or more records per segment. (Up to 25 segments.)
- ▶ **Descriptor record:** Contains 1 to 25 segment descriptors. One record.
- ▶ **Segment ID record:** Contains 1 to 25 segment IDs. One or more records.
- ▶ **Data segments:** One or more records per segment. (Up to 25 segments.)



# SPK File Structure - Description

Navigation and Ancillary Information Facility

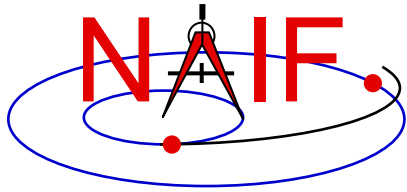
- **File record**
  - **Contents**
    - » **Internal file name (set by file creator)**
    - » **Architecture and binary file format identifiers**
    - » **File structure parameters**
    - » **FTP transmission corruption detection string**
  - **Used by SPK reader and writer subroutines**
- **Comment Area**
  - **A place where “metadata” (data about data) may be placed to help a user of the SPK file understand the circumstances of its production and any recommendations about for what uses it was intended**
- **Descriptor record and Segment ID record**
  - **One of each of these is needed for every collection of 1-to-25 segments**
- **Segment[s]**
  - **Collection[s] of ephemeris data**
    - » **Minimum of one segment**
    - » **Maximum:**
      - **The practical maximum is a few thousand segments**
      - **Serious performance degradation occurs above 100000 segments for a single body**
      - **Absolute limits are imposed by the range of the INTEGER data type for your computer**
  - **Numerous SPK types may be used within an SPK file, but only one SPK type may appear within a given segment**
  - **Segments of different types may be intermixed within a given SPK file**



# What is an SPK Segment?

Navigation and Ancillary Information Facility

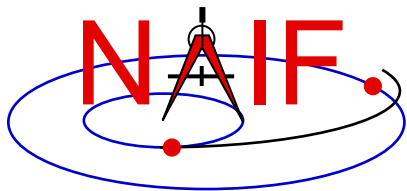
- **A segment is a collection of information:**
  - » providing ephemeris (position and velocity) of a single object ...
  - » given relative to a single center of motion ...
  - » specified in a single reference frame known to SPICE ...
    - Either built-in (“hard coded”) or defined in a loaded frames kernel (FK)
  - » covering a specified, continuous period of time, and
  - » using a single SPK data type.
- Example: ephemeris for the Voyager 2 spacecraft, relative to the center of the Neptunian system (Neptune’s barycenter), given in the J2000 inertial reference frame, covering a specific period of time, and using the Hermite interpolation with variable length intervals SPK type (type 13)
- **An SPK segment must contain enough data to yield an object’s state at any epoch within the time bounds associated with the segment**
  - This has implications that depend on the SPK type being produced



---

Navigation and Ancillary Information Facility

# Discussion applicable to all SPK production methods

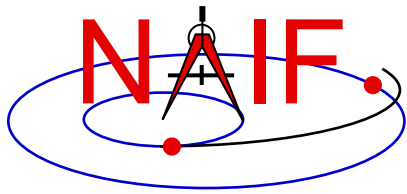


# The SPK Family

## Navigation and Ancillary Information Facility

Type	Description	Notes
1	Modified divided difference arrays	Unique form produced at JPL; not likely to be useful to others.
2	Chebyshev polynomials for position; fixed length time intervals.	Velocity is obtained by differentiation. Used at JPL for planets. Evaluates quickly.
3	Chebyshev polynomials for position and velocity; fixed length time intervals	Separate polynomial sets for position and velocity. Used at JPL for natural satellites.
4	Special form used only by Hubble Space Telescope	Not available for general use.
5	Discrete states using weighted two-body propagation	Ok if motion very closely approximates two-body motion.
6	Special form of trigonometric expansion of elements for Phobos and Deimos	Not available for general use.
7	Precessing elements	Not available for general use.
8	Lagrange interpolation of position and velocity; fixed length intervals between states	Use Type 9 unless state spacing is truly uniform when measured in the TDB system.
9	Lagrange interpolation of position and velocity; variable length intervals between states	Versatile type; easy to use with MKSPK.
10	Weighted two-line element sets (Space Command)	Handles both "near-earth" and "deep space" versions.
11	Not used	
12	Hermite interpolation of position and velocity; fixed length intervals between states	Use Type 13 unless state spacing is truly uniform when measured in the TDB system.
13	Hermite interpolation of position and velocity; variable length intervals between states	Versatile type; easy to use with MKSPK. Use for DSN support.
14	Chebyshev polynomials for position and velocity, variable length time intervals	The most flexible of the Chebyshev types.
15	Precessing conic elements propagator	---
16	Special form used by ESA's Infrared Space Observatory	Not available for general use.
17	Equinoctial elements	Used for some satellites.
18	Emulation of ESOC's "DDID" format	Used for SMART-1, MEX, VEX, and Rosetta
19	Revised emulation of ESOC's "DDID" format	Similar to Type 18, but uses "mini-segments" to reduce the number of true segments needed. (New in N65 toolkits.)
20	Chebyshev polynomials for velocity; fixed length time intervals.	Provided to handle ephemerides produced by the Institute of Applied Astronomy in St. Petersburg, Russia. (New in N65 toolkits.)
21	Modified divided difference arrays	Same as Type 1 except allows greater number of coefficients.

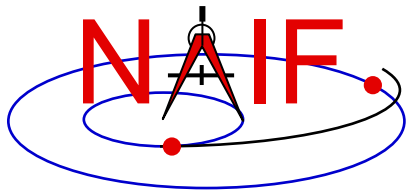




# Recommended SPK Data Types - 1

Navigation and Ancillary Information Facility

- **SPK type 2 (Chebyshev polynomials for position, velocity given by differentiation)** Used at JPL for planetary ephemerides.
- **SPK type 3 (Separate Chebyshev polynomials for position and velocity)** Used at JPL for satellite ephemerides.
- **SPK type 5 (Weighted two-body extrapolation)** Often used for comets and asteroids, as well as for sparse data sets where a two-body approximation is acceptable.
- **SPK types 9 and 13 (Sliding-window Lagrange and Hermite interpolation of unequally-spaced states)** Often used by users of NAIF's "Make SPK" (MKSPK) application.
- **SPK type 10 (weighted Space Command two-line element extrapolation)** Often used for earth orbiters.
- **SPK type 14 (Separate Chebyshev polynomials for position and velocity, with variable time steps)** This is the most flexible Chebyshev data type.
- **SPK type 15 (Precessing conic elements)** Provides a very compact ephemeris representation; limited to orbits where this type of approximation is valid.
- **SPK type 17 (Equinoctial elements)** Most suited for representation of ephemerides of natural satellites in equatorial or near-equatorial orbits.

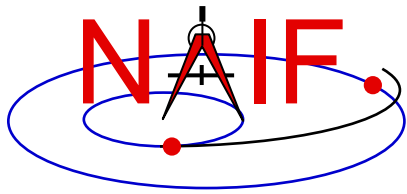


# Recommended SPK Data Types - 2

---

Navigation and Ancillary Information Facility

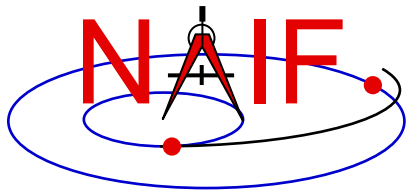
- **Each type has certain properties that may promote or limit its usefulness in a particular application. These properties include, but are not limited to the following.**
  - » **Ability to model the actual ephemeris to be represented with the accuracy required for your application.**
  - » **Consistency between velocity and derivative of position.**
  - » **Evaluation speed (performance).**
  - » **Compactness (file size).**
  - » **Availability of SPICE software needed to write files of that type.**
- **Users are encouraged to consult with NAIF about the suitability of an SPK type for a particular purpose.**



# Creating Multiple SPK Segments

Navigation and Ancillary Information Facility

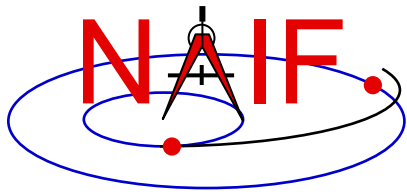
- **Each SPK segment must have a single object, center of motion, reference frame and SPK data type.**
- **Limiting segment size to 10,000 states or “packets of ephemeris data” can improve performance when searching within a segment.**
  - Absolute limits on segment size depend on the size of the INTEGER data type.
- **The total number of segments for any one body in a single SPK file must be kept under the dimension of the SPKBSR segment buffer, currently set to 100,000.**
  - The total number of segments for any one body, contributed by all loaded SPK files, must be kept under the 100K limit.
  - For best efficiency, the total number of concurrently loaded segments for all bodies should be less than this 100K limit. Adherence to this criterion ensures that loaded SPK files need not be searched repeatedly for segment descriptors (which contain summary data).
  - More details about reading efficiency are provided at the end of this tutorial.
- **One may elect to initiate a new segment (or more) as the means for modeling a propulsive maneuver.**
  - This is because the SPK reader modules will NOT allow interpolation over a segment boundary.
- **When starting a new segment you may use a new segment identifier, for instance to indicate a new trajectory leg after a maneuver.**
  - Can only be done if using SPK writer modules—not if using the MKSPK application.



# Choosing Polynomial Degree

Navigation and Ancillary Information Facility

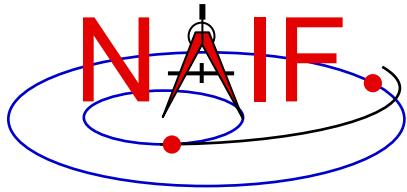
- **If you make a type 8 or 9 (Lagrange interpolation) or a type 12 or 13 (Hermite interpolation) SPK file you must specify the degree of the interpolating polynomial that the SPK reader subroutine will use.**
  - This choice needs some consideration about desired accuracy, file size and evaluation speed (performance).
  - This choice is also affected by the “smoothness” of the orbit data you wish to represent with an SPK file.
  - The allowed range of degree is 1-to-27. In addition, to ensure position and velocity continuity over the time span covered by the orbit data:
    - » for types 8 and 9, the polynomial degree must be odd.
    - » for types 12 and 13, the polynomial degree must be equivalent to 3-mod-4, meaning degree 3, 7, 11, 15, 19, 23 or 27.



# Reference Frame

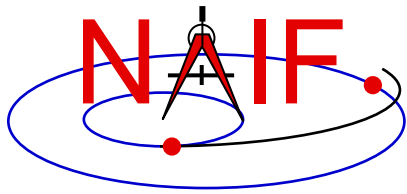
Navigation and Ancillary Information Facility

- **Any reference frame known to the SPICE system, whether built-in (hard coded) or defined at run time through a Frames Kernel (FK), may be used for ephemeris data placed in an SPK file.**
  - Exception: don't use a “two-vector” dynamic frame type
- **Some examples of typical reference frames used:**
  - Inertial (non-rotating):
    - » J2000: (Earth Mean Equator and Equinox of J2000, a.k.a. EME2000)
    - » ECLIPJ2000: (Mean ecliptic and equinox of J2000)
  - Body-fixed (non-inertial)
    - » ITRF93: (International Terrestrial Reference Frame, for earth)
    - » MOON\_PA: (MOON\_Principal Axes)



Navigation and Ancillary Information Facility

# SPK Production Methods



# Choices for Making an SPK File

---

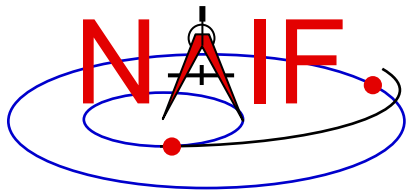
Navigation and Ancillary Information Facility

- **There are two\* methods available for making an SPK file.**
  1. **Take a data file produced by your own trajectory propagator program and input this into the conversion utility (MKSPK) provided by NAIF that outputs an SPK file.**
  2. **Incorporate the appropriate SPK writer modules (subroutines) into your own code.**
    - » **Add these routines to your trajectory estimator/ propagator.**

**or...**

  - » **Write your own “post-processor” conversion utility, similar to MKSPK mentioned above.**
- **Both methods are described in the next few pages.**

**\*The OEM2SPK and NIOSPK specialized utilities are also methods, but are not discussed in this tutorial.**



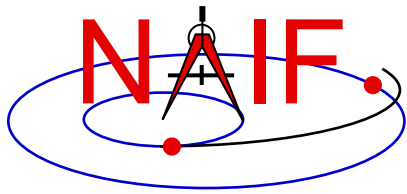
# Making Your Choice - 1

---

Navigation and Ancillary Information Facility

- **Using the MKSPK program provided in the Toolkit could be easiest for many situations.**
  - **Provides considerable flexibility for accepting a wide assortment of input data types and formats.**
  - **Allows one to make multi-segment SPK files when the target, center of motion, reference frame, or SPK type changes. (But not as straight forward as it could/should be.)**
    - » **Best done through multiple program executions (although one could be tricky and accomplish this in a single execution).**
    - » **A future version of MKSPK may better accommodate this.**
    - » **Note: production of multiple segments in type 5, 8, 9, 12 and 13 SPK files, when the amount of input data requires this, is automatically handled.**



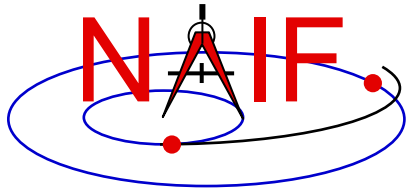


# Making Your Choice - 2

---

Navigation and Ancillary Information Facility

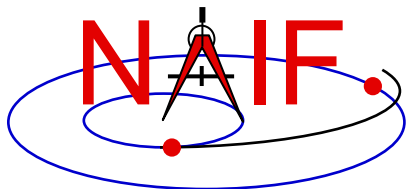
- **Using the SPK “writer” modules found in SPICELIB, CSPICE, Icy and Mice offers the greatest flexibility and user control.**
  - Using these requires that you write your own program.
  - You’ll likely need to use some additional SPICE modules as well.



---

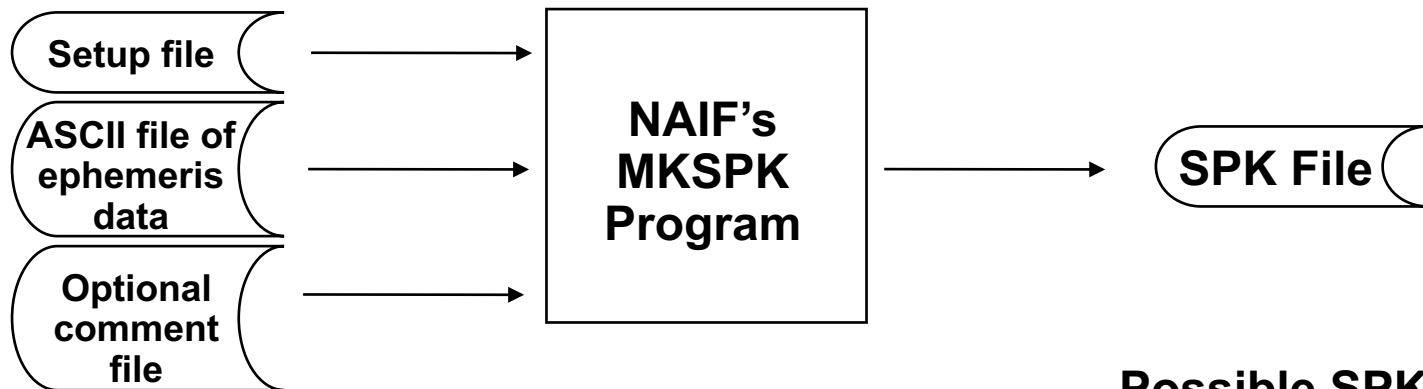
Navigation and Ancillary Information Facility

# Using NAIF's MKSPK Application Program



# Using the MKSPK Utility - 1

Navigation and Ancillary Information Facility

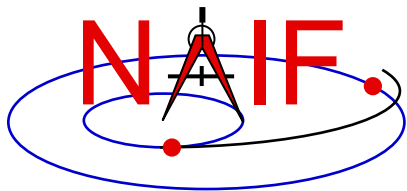


**Suitable kinds of input ephemeris data are:**

- Table of Cartesian state vectors
- Table of conic elements
- One or more sets of equinoctial elements
- One or more sets of Space Command two-line elements

**Possible SPK data types produced are:**

- Type 05
- Type 08
- Type 09
- Type 10
- Type 12
- Type 13
- Type 15
- Type 17



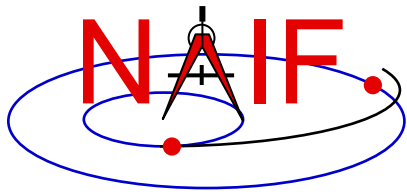
# Using the MKSPK Utility - 2

Navigation and Ancillary Information Facility

**This table indicates which SPK types can be made from the four kinds of input data accepted by MKSPK**

SPK Type Produced by MKSPK -->	5	8	9	10	12	13	15	17
<b>Input Data Type</b>								
Cartesian state vectors	Y	Y	Y	N	Y	Y	Y	Y
Conic elements	Y	Y	Y	N	Y	Y	Y	Y
Equinoctial elements	N	N	N	N	N	N	N	Y
Space Command Two-line elements	N	N	N	Y	N	N	N	N

**Y = Yes    N = No**

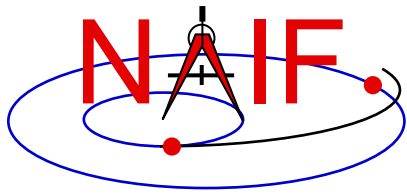


# Using the MKSPK Utility - 3

---

Navigation and Ancillary Information Facility

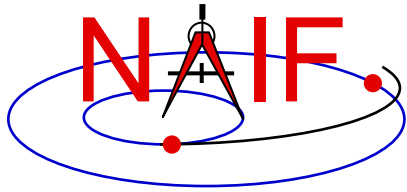
- **MKSPK will produce a file consisting of one or more segments as needed.**
  - It can write up to 10,000 data points in one segment.
  - For multi-segment files based on types 5, 8, 9, 12 and 13, the program will repeat sufficient data points at both sides of each interior segment boundary to ensure the SPK file will provide a continuous ephemeris through the segment boundary epoch.
- **You can use MKSPK to add a new segment to an existing SPK file.**
- **You can use SPKMERGE to merge two or more SPK files made from separate executions of MKSPK.**
  - It's important to fully understand how SPKMERGE works if you do this; see the User's Guide.



# Using the MKSPK Utility - 4

Navigation and Ancillary Information Facility

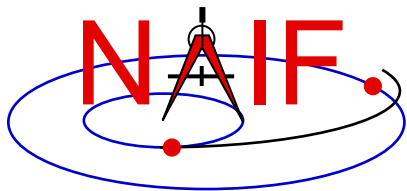
- **MKSPK does not provide direct/specific means for including propulsive maneuvers within an SPK file.**
  - Instead, use either of these two methods.
    - » **Append a new SPK segment to an existing SPK file, using MKSPK.**
    - » **Merge a collection of SPK files, using SPKMERGE.**



---

Navigation and Ancillary Information Facility

# Using SPK “Writer” Modules



# Using SPK Writer Routines

Navigation and Ancillary Information Facility

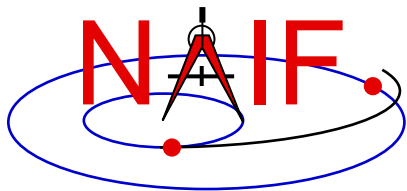
- The next several charts outline how to use the “SPK writer” modules available in the Toolkit libraries. The types available in the supported Toolkits are summarized below.\*

SPK Types Supported	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
SPICELIB (Fortran 77)	✓	✓	✓	■	✓	■	■	✓	✓	✓	■	✓	✓	✓	✓	■	✓	✓	✓	✓	✓
CSPICE (C)		✓	✓	■	✓	■	■	✓	✓	✓	■	✓	✓		✓	■	✓	✓			✓
ICY (IDL)		✓	✓	■	✓	■	■	✓	✓	✓	■	✓	✓			■	✓				
MICE (MATLAB)				■		■	■	✓			■					■					

■ = 4, 6, 7, 16: made for special circumstances; not available for general use  
 ■ = 11: ID was not used

\* As of Toolkit version N66





# What Routines To Use - 1

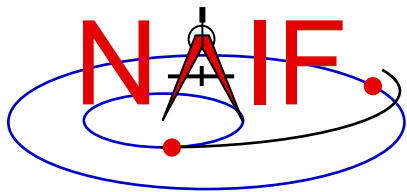
Navigation and Ancillary Information Facility

## For all except SPK type 14

<b>SPKOPN</b>	Open a new SPK file. (Use SPKOPA to append to existing file.)
<b>SPKWxx</b>	Write a segment of SPK type xx
.	
.	
<b>[SPKWxx]</b>	[ Write more segments ]
.	[ Repeat as needed ]
.	
<b>SPKCLS</b>	Close the file

**[ ... ] indicates possible multiple occurrences**

These routine names are for the FORTRAN (SPICELIB) Toolkit. For CSPICE the names are the same but are in lower case and have an "\_c" appended. For Icy and Mice, module names are case-insensitive and have "cspice\_" prepended.



# What Routines To Use - 2

Navigation and Ancillary Information Facility

## For SPK type 14

**SPKOPN, SPKOPA**

Open file to add data

**SPK14B**

Begin a new segment

**SPK14A**

Add data to segment

**[SPK14A]**

Add more data

**SPK14E**

End the segment

**SPK14B**

Begin a new segment

**SPK14A**

Add data to segment

**[SPK14A]**

Add more data

**SPK14E**

End the segment

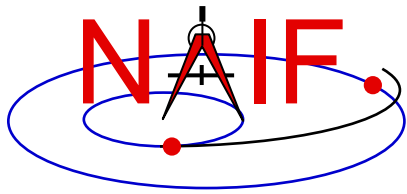
etc.

**SPKCLS**

Close the file

Repeat  
as needed

**[SPK14A]** indicates possible multiple occurrences

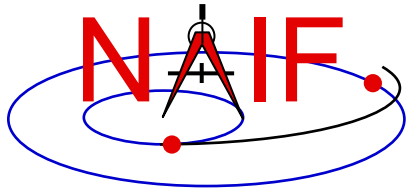


# Close the SPK File

---

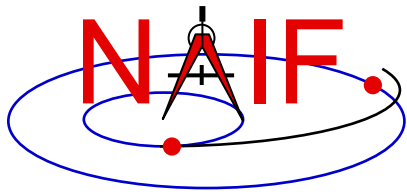
Navigation and Ancillary Information Facility

- **Once you have completed the addition of all data to your SPK file, be sure to call the SPKCLS routine to close the file.**
  - Failure to properly close an SPK file will result in a problem file having been produced.
- **This point is emphasized here because it has been a frequent problem.**



Navigation and Ancillary Information Facility

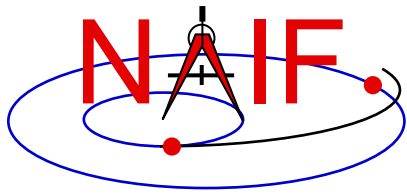
# Finishing Up



# Not Quite Done Yet

Navigation and Ancillary Information Facility

- **You've now used either MKSPK or the appropriate SPK writer routines to produce an SPK file. To complete the job you should do the following.**
  - **Add comments (metadata) to the comment area of the SPK file.**
    - » **This could have been done during execution of MKSPK.**
    - » **It can be done after the SPK has been created by using the Toolkit's "commnt" utility program.**
  - **Validate the file before sending it off to your customer.**
  - **Consider if there is a need to merge this newly made SPK file with others.**
- **See the next several charts for more information on these subjects.**

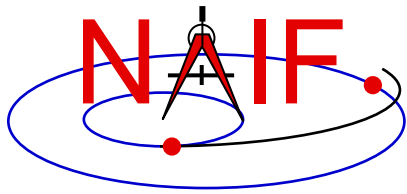


# Add Comments (metadata)

---

Navigation and Ancillary Information Facility

- **It is strongly recommended that the producer of an SPK file add to the file, in the “comment area,” appropriate descriptive information. Topics should include at least:**
  - when, how and by whom the file was created,
  - intended use for the file,
  - cautions or restrictions on how the file is to be used.
- **The comments might also include some of these topics.**
  - Time coverage
  - Ephemeris objects included
  - Type(s) of data used (in the sense of reconstructed versus predicted)
  - Any available estimates of accuracy
  - Sources of the data used to produce this SPK file
  - Name(s) of previously generated SPK file(s) being replaced by this file
  - Any knowledge of plans for future updates to (replacements for) this file
  - Name and version number of your SPK production program
  - Type of platform (hardware/OS/compiler) on which the SPK file was generated
- **If you are modifying an existing SPK file, be sure to check the existing comments to see if updates are needed in addition to adding appropriate new ones.**

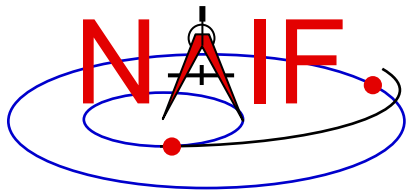


# How to Add Comments to an SPK

---

Navigation and Ancillary Information Facility

- **Several means are available for adding comments (metadata) to an SPK file.**
  - An option in the MKSPK program allows comments supplied in a separate text file to be added to the comment area during MKSPK execution.
  - Use the “COMMNT” utility program from the SPICE Toolkit.
    - » This may be run as an interactive program or in command line mode within a script.
  - If using FORTRAN, C or IDL you can use APIs.
    - » Not currently supported in MATLAB.

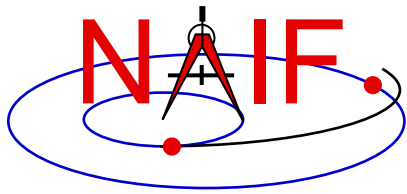


# Validate the SPK File

Navigation and Ancillary Information Facility

- **Validation of SPK files is critical**
  - **Caution is needed more for one-of-a-kind files than for those generated in a previously tested, unchanging process.**
  - **Some SPICE utility programs might help with your validation.**
    - » **SPY: can do a variety of structure and semantic checks.**
      - **SPY is available from the Utilities link of the NAIF website.**
    - » **SPKDIFF: used to statistically compare two supposedly similar SPK files.**
      - **SPKDIFF is available in each Toolkit package and also from the Utilities link of the NAIF website.**
  - **Consider writing your own validation program.**
  - **Caution: successfully running an SPK summary program (e.g. BRIEF or SPACIT) or successfully running the format conversion program (TOXFR or SPACIT) is a positive sign, but is not a sufficient test.**

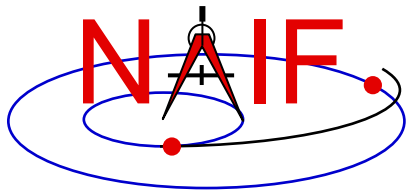




# Validate the Overall Process

Navigation and Ancillary Information Facility

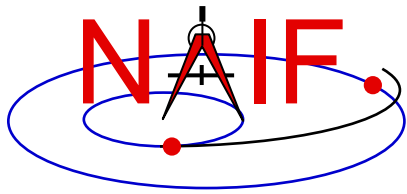
- **When you first start producing SPK files, or when changing the SPK “type” used or the kind of mission (trajectory) to be represented, validation (or revalidation) of the overall process is advised.**
  - Validation of not only SPKs, but of end products derived from SPKs, is advised.
- **Consider writing a program that compares states from your source data with states extracted from your new SPK file.**
  - Do this using **interpolated states** from your source data—not only the states placed in the SPK file.
  - Verify a uniformly good fit on the whole time interval covered by the file.



# Common SPK Production Errors

Navigation and Ancillary Information Facility

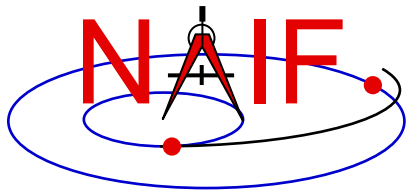
- **Some of the most common SPK production errors are these**
  - Picking an SPK type that can not well represent the ephemeris data you wish to represent
  - For Types 8, 9, 12 and 13, picking an invalid polynomial degree
  - For Types 8, 9, 12 and 13, picking a polynomial degree that does not allow high fidelity representation of your source ephemeris
  - For Types 8, 9, 12 and 13, having a HUGE change in time step size in two time-adjacent SPK segments
  - For Types 8, 9, 12 and 13, allowing position and/or velocity data to have unduly large discontinuities at segment boundaries
  - Leaving time gaps between segments (even very small, sub-second gaps can cause problems for users)
- **It takes careful design to avoid, and careful validation to detect some of these!**



# Make a Merged SPK File ?

Navigation and Ancillary Information Facility

- Sometimes it is helpful to customers if portions of two or more SPK files are merged into just one.
  - (Sometimes the opposite is true, so be careful!)
- If making a merged product is appropriate, use the SPICE utility SPKMERGE.
  - Read the SPKMERGE User's Guide.
    - » Be especially aware of how SPKMERGE directives affect the **precedence order** of the data being merged. (This is different from the precedence order that applies when one reads an SPK file or files.)
  - Carefully examine your results (probably using either BRIEF or SPACIT) to help verify you got what you expected.
- If you've made a merged SPK file, check to see that the included comments are appropriate.

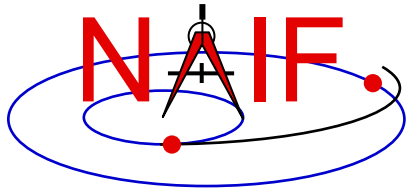


# Get Help

---

Navigation and Ancillary Information Facility

- **Ask JPL's NAIF team for assistance with:**
  - picking the SPK type to be used
  - picking the method for producing SPK files
  - designing tests to validate the process

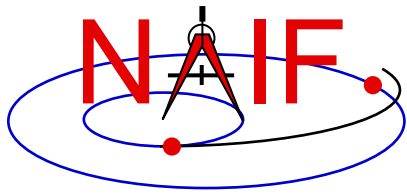


---

Navigation and Ancillary Information Facility

## Backup

- **Making SPKs for use by the DSN**
- **SPK reading efficiency**

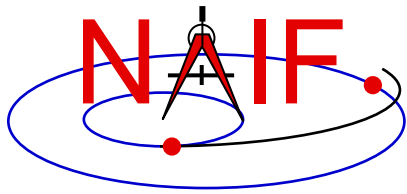


# DSN Interface Overview

---

Navigation and Ancillary Information Facility

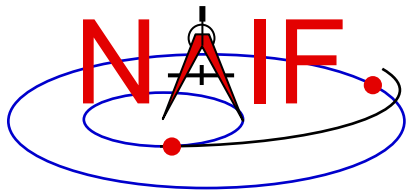
- **SPKs prepared for use in the DSN/SPS may be used in one or more of four software sets:**
  - **Metric Predicts Generator (MPG)**
    - » Used for view period generation, DSN scheduling and DSN metric predicts (antenna pointing and tuning of the transmitters and receivers)
  - **Telecomm Predicts (UTP/TFP)**
    - » Subsystem for prediction and analysis of telecommunications signal levels
  - **Radiometric Modeling and Calibration Subsystem (RMC)**
    - » Used to calibrate atmospheric effects on radio waves
  - **Delta Differenced One-way Range (Delta-DOR) subsystem**
    - » A special tracking data type providing additional precision to spacecraft navigation
- **All SPKs delivered to the SPS must pass through a front-end gateway that has some restrictions.**



# SPS Validation Gateway

Navigation and Ancillary Information Facility

- **The SPS' front-end gateway requires:**
  - an SPK contain data for only one spacecraft
    - » The presence of non-spacecraft ephemeris data is ok
  - an SPK have no data gaps for the spacecraft
- **Additionally, for historical reasons, the spacecraft SPK must be of Type 1 or Type 13**
  - This restriction could be diminished or removed sometime in the future if DSN personnel find time to test the Metric Predicts Generator code using additional SPK data types.



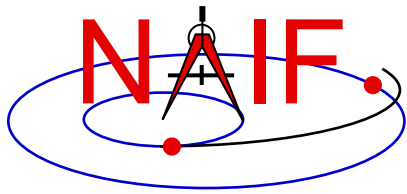
# SPK Reading: Efficiency Issues - 1

---

Navigation and Ancillary Information Facility

- **SPK file creators should design files to support efficient read access by those who will use the SPKs.**
  - This requires knowledge of how SPK file attributes impact efficiency.
- **When you store "large" amounts ( $>10^7$  states or data packets) of ephemeris data in one or more SPK files, SPK reading efficiency may be affected by:**
  - SPK segment size
  - the number of segments for a body in one SPK file
  - the number of segments for a body contributed by multiple SPK files
  - the number of loaded segments for all bodies
  - the number of loaded files





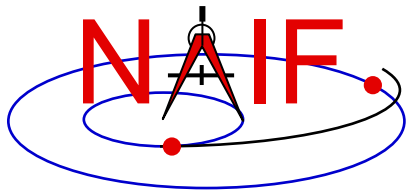
# SPK Reading: Efficiency Issues - 2

---

Navigation and Ancillary Information Facility

- **Segment size**

- **When a segment contains more than 10,000 states or data packets, the SPK readers will generally take longer to search the segment for requested data.**
  - » **When the segment is larger than this size, more records are read to look up segment directory information. If these records are not buffered, more physical records are read from the SPK file.**
- **There is a trade-off between segment size and numbers of segments and files.**
  - » **It can be preferable to have large segments rather than have "too many" segments or files. (Read on)**

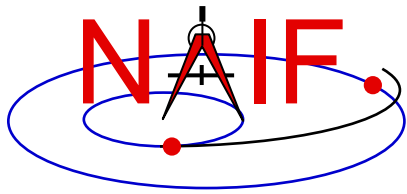


# SPK Reading: Efficiency Issues - 3

---

Navigation and Ancillary Information Facility

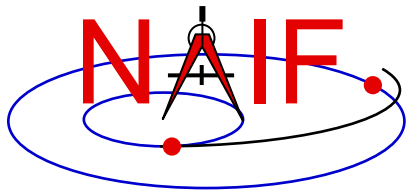
- **Number of segments for a body in one SPK file**
  - **An SPK file MUST not contain more segments for one body than can be "buffered" at one time.**
    - » **The SPK reading system buffers coverage descriptions ("segment descriptors") for segments it has examined to satisfy previous requests for state data.**
      - **Don't confuse descriptor buffering with data buffering.**
        - **The SPK reading system also buffers segment DATA, as opposed to segment descriptors, but this is not relevant to this discussion.**
    - » **One fixed-size buffer is used for all SPK segment descriptors.**
      - **The size of this buffer is given by the parameter "STSIZE," declared in the SPKBSR suite of routines.**
      - **STSIZE is currently set by NAIF to 100,000.**
        - **NAIF recommends that users NOT change this parameter since maintenance problems may result.**
    - » **Unsurprisingly, the system works best when all needed segment descriptors are buffered simultaneously.**



# SPK Reading: Efficiency Issues - 4

Navigation and Ancillary Information Facility

- **Number of segments for a body in one SPK file, continued.**
  - » The buffering scheme is "lazy": no descriptors are buffered for segments that haven't been examined.
    - But when an SPK file is searched for data for a specified body, descriptor data for ALL segments in the file for that body are buffered.
  - » The buffering algorithm can "make room" in the buffer by discarding unneeded, buffered descriptor data.
    - A "least cost" algorithm decides which buffered data to discard.
  - » When more buffer room is needed than can be found:
    - The SPK reading system reads data directly from SPK files without buffering descriptor information.
    - This is NOT an error case: the SPK system will continue to provide correct answers. **But the system will run VERY SLOWLY.**
      - This situation is analogous to "thrashing" in a virtual-memory operating system.
      - If buffer overflow occurs frequently, the SPK reading system may be too slow to be of practical use.

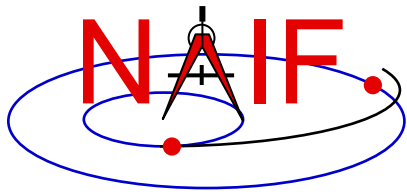


# SPK Reading: Efficiency Issues - 5

---

Navigation and Ancillary Information Facility

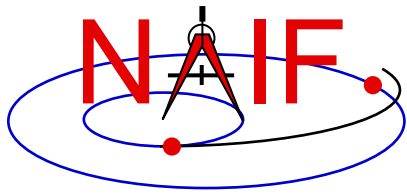
- **Number of segments for a body contributed by multiple SPK files:**
  - **Buffer overflow can occur if too many segments for one body are contributed by multiple loaded SPK files.**
    - » **Overflow can take longer to occur than in the single-SPK case, due to lazy buffering: files that haven't been searched don't consume buffer space.**
      - Thus an impending overflow problem may not be detected early in a program run.
  - **User applications can avoid buffer overflow if data are appropriately spread across multiple SPK files.**
    - » **Applications can avoid buffer overflow by:**
      - loading only those files of immediate interest
      - unloading files once they're no longer needed



# SPK Reading: Efficiency Issues - 6

Navigation and Ancillary Information Facility

- **Number of segments for all bodies, contributed by all loaded SPK files:**
  - Buffer overflow does not result from loading SPK files contributing more than STSIZE segments for different bodies.
  - However, if the total number of loaded segments for bodies of interest exceeds STSIZE, thrashing can occur as descriptor data are repeatedly discarded from the buffer and then re-read.
    - » Loaded segments for bodies for which data are not requested do not contribute to the problem.
  - For best efficiency, load only files contributing fewer than a total of STSIZE segments for all bodies of interest.
    - » When more than STSIZE segments are needed, applications should process data in batches: unload files containing unneeded data in order to make room for new files.

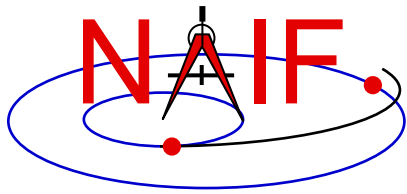


# SPK Reading: Efficiency Issues - 7

---

Navigation and Ancillary Information Facility

- **Number of loaded SPK files:**
  - Up to 5000 SPK files may be loaded at one time by an application.
    - » The 5000 limit applies to DAF-based files, so loaded C-kernels and binary PCK kernels count against this limit.
  - But loading large numbers of SPK files hurts efficiency:
    - » Since operating systems usually allow a process to open much fewer than 5000 files, the SPK system must open and close files via the host system's file I/O API in order to provide a "virtual" view of 5000 open files.
      - The more such file I/O, the slower an application runs.
    - » Loading a large number of SPK files could result in a buffering problem if too many segments are loaded for the bodies of interest.



# SPK Reading: Efficiency Issues - 8

Navigation and Ancillary Information Facility

- **Recommendations**

- **Limit segment counts to avoid buffer overflow and thrashing**
  - » **Never have more than STSIZE segments for one body in an SPK file and never have more than STSIZE segments for one body loaded simultaneously.**
  - » **Don't require users to have more than STSIZE segments loaded at one time.**
  - » **If necessary, use larger segments to enable smaller segment counts.**
- **Consider distributing SPK data across multiple files ...**
  - » **so as to make selective SPK loading convenient**
    - **facilitate processing data in batches**
  - » **so that loading very large numbers of SPK files at once is unnecessary**