The SPICE Ephemeris Subsystem SPK

Emphasis on reading SPK files

March 2006

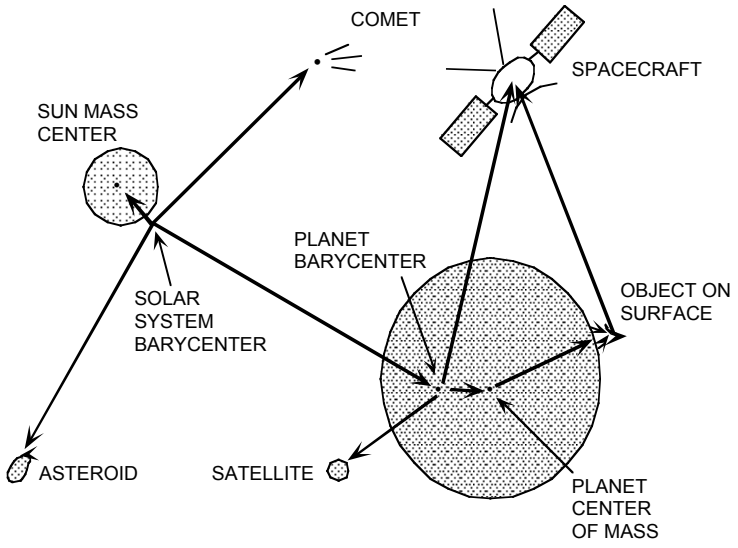
Navigation and Ancillary Information Facility

SPK File Contents

- An SPK file holds ephemeris data for any number/types of solar system objects
  - "Ephemeris data" means position and velocity of one object relative to another
  - "Solar system object" means any spacecraft, planet (mass center or barycenter), satellite, comet, asteroid, sun and the solar system barycenter. Can also be a lander or rover or Earth station (e.g. DSN station). Can also be a designated point on an object, such as the top of a mast on a lander or the location of an antenna feed on a spacecraft. Could even be a Lagrange point.
- An SPK file can hold data for one object, or for any combination of objects
  - Examples:
    - » The Cassini orbiter
    - » The Cassini orbiter and the Huygens probe
    - » The Mars Odyssey and Mars Express orbiters, Mars, Phobos, earth, sun, and the MER rovers

Examples of SPICE Ephemeris Objects

Navigation and Ancillary Information Facility



Planets and Planet Systems

Navigation and Ancillary Information Facility

- **Planets and their satellites orbit the planet system barycenters**
  - **For example, the Jupiter mass center (599) and each of Jupiter's satellites (501 - 5xx) orbit the Jupiter system barycenter (5)**
- **Planet system barycenters (i.e. 1 through 9) and the sun (10) orbit the solar system barycenter (0)**
- **With no satellites, Mercury's and Venus' barycenters (1 and 2) occupy the same locations as their mass centers (199 and 299)**
- **Because the masses of Phobos and Deimos are so small compared to the mass of Mars, the mass center for Mars (499) is treated as equivalent to the Mars barycenter (4)**

Barycenters and Mass Centers

Navigation and Ancillary Information Facility

Body Mass Center	System Barycenter	Body center offset from Barycenter (km)*	Offset as % of body radius*
Sun (10)	SSB (0)	1,295,728	185%
Mercury (199)	M. BC (1)	0	0
Venus (299)	V. BC (2)	0	0
Earth (399)	E. BC (3)	4874	76%
Mars (499)	M. BC (4)	~0	~0
Jupiter (599)	J. BC (5)	165	0.2%
Saturn (699)	S. BC (6)	296	0.5%
Uranus (799)	U. BC (7)	17	0.06%
Neptune (899)	N. BC (8)	74	0.3%
Pluto (999)	P. BC (9)	1500	125%

* At epoch 1997 JAN 01

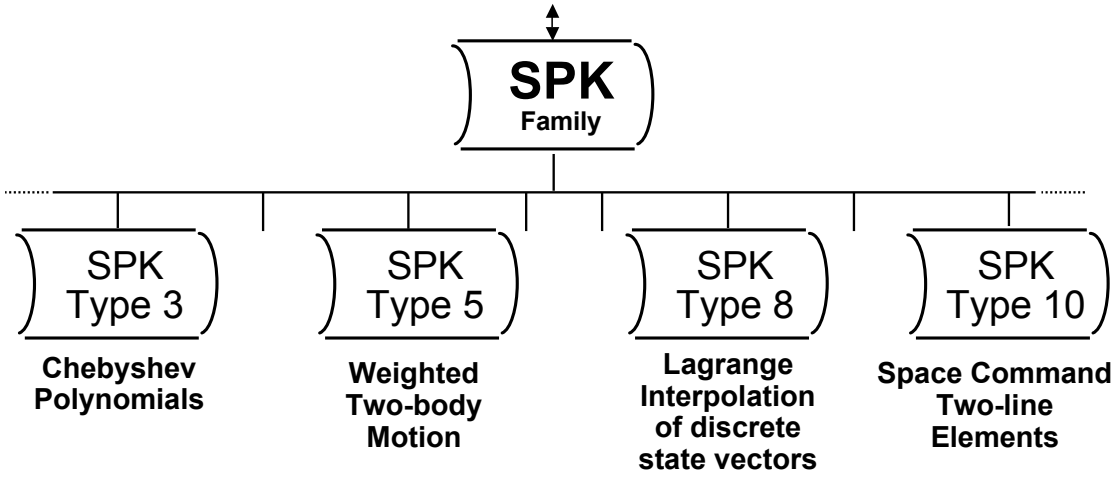
SPK Data Type Concept

Navigation and Ancillary Information Facility

SPK files contain various mathematical representations of trajectory data ("data types"), but the high-level user interface (SPK "readers") is type-independent:

CALL SPKEZR ('target', time, 'frame', 'correction', 'observer', state, light_time)

CALL SPKPOS ('target', time, 'frame', 'correction', 'observer', positn, light_time)



Widely Used SPK Data Types

Navigation and Ancillary Information Facility

- SPK type 1 (Modified divided difference arrays) is the representation used by JPL orbit determination software for spacecraft ephemerides.
- SPK type 2 (Chebyshev polynomials for position, velocity given by differentiation) is used for JPL planetary ephemerides.
- SPK type 3 (Separate Chebyshev polynomials for position and velocity) is used for JPL satellite ephemerides.
- SPK type 5 (Weighted two-body extrapolation) is often used for comets and asteroids, as well as for sparse data sets where a two-body approximation is acceptable.
- SPK type 10 (Space command two-line elements) is often used for earth orbiters.
- SPK types 9 and 13 (Sliding-window Lagrange and Hermite interpolation of unequally-spaced states) are often used by non-JPL ephemeris producers and by MKSPK users.

Why Have Multiple Data Types?

Navigation and Ancillary Information Facility

- To replicate data originally provided in other formats
  - Types 1, 2, 3, 8, 10, 14, 15, 17 and 18 were developed to enable accurate duplication of data from ephemeris developers.
- To allow SPK producers to choose an ephemeris representation well-suited for their applications. For example:
  - Weighted two-body extrapolation (type 5) yields compact files; may be used with sparse data. Only accurate for motion that is well approximated by the two-body model.
  - SPK files based on sliding-window Lagrange and Hermite interpolation (types 9 and 13) are easy to create. Position can be made arbitrarily accurate with sufficiently small time separation of states.
  - Chebyshev polynomials (types 2, 3, 14) yield best combination of evaluation speed and accuracy. File creator must do more work to use these data types.

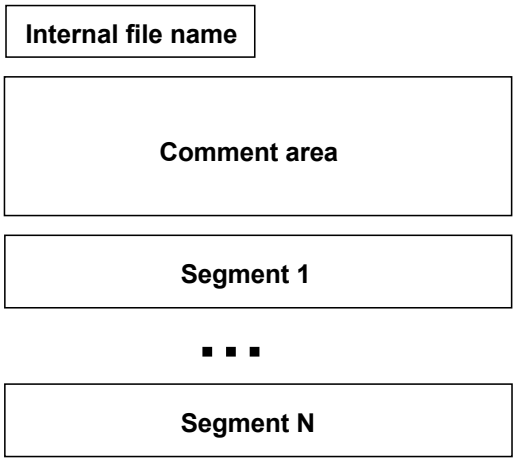
SPK File Organization

Navigation and Ancillary Information Facility

- An SPK file is made up of one or more segments.
  - Segment: data for one object, relative to one observer (center of motion), in one reference frame, for a specified time interval.
    - » Each segment is implemented using one data type. Different segments in a file may use different data types.
- When writing an SPK file
  - A new segment must be started when any of the items mentioned above changes.
  - A new segment may be started whenever the SPK producer elects to do so.
- The SPK readers SPKEZR and SPKPOS return states or position vectors for any time between a segment's start and stop time (even when the segment contains discrete data).
  - Achieved through interpolation or using analytical function, depending on the SPK type.
- By definition, interpolation across segments or beyond the bounds of a segment is not permitted.

SPK File Structure: Reader's View

Navigation and Ancillary Information Facility

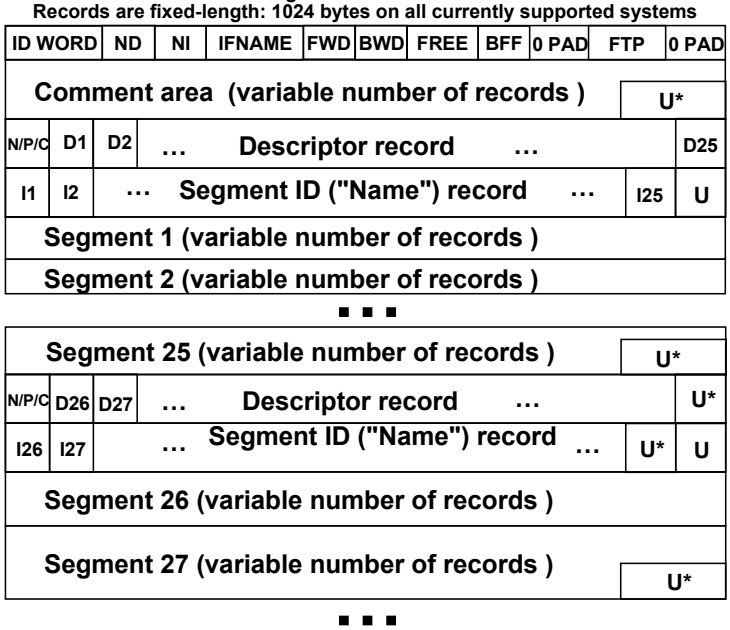


SPK Physical File Structure

Navigation and Ancillary Information Facility

Example: SPK file with 27 or more segments

Diagram not to scale.
Records are fixed-length: 1024 bytes on all currently supported systems



File record: always present.

The comment area may be empty.

An SPK file has at least one descriptor record and one segment ID ("name") record. These records contain up to 25 pairs of segment descriptors and segment IDs.

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

DN: Descriptor for segment N

IN: Segment ID for segment N

N/P/C: Next, previous record pointers and descriptor count

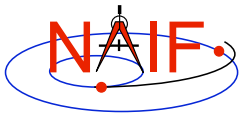
U: Unused space

U*: Possibly unused space

SPK Segment Order and Priority

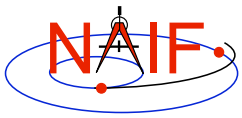
Navigation and Ancillary Information Facility

- SPK segments need not be ordered according to time or body.
- Segment order does imply priority: when two segments both contain data that satisfy a data request, the segment located later in the file is selected.
  - In the coverage overlap time interval, the lower priority segment is invisible.
- Well-constructed SPK files should not have segment coverage overlaps except at segment coverage boundaries.
- SPKMERGE can be used to remove coverage overlaps from an SPK file.



Retrieving State Vectors:
Kernel Data Needed

- To retrieve state vectors (position and velocity) of "objects" (spacecraft, planets, satellites, sun, comets, asteroids) from an SPK file one normally needs two kinds of SPICE kernels
  - Ephemeris kernel(s) (SPK)
  - Leap seconds kernel (LSK)
    - » Used to convert between Coordinated Universal Time (UTC) and Ephemeris Time (ET)



Retrieving State Vectors:
Deriving States

- The SPICE SPK subsystem looks up ephemeris data as needed to satisfy a state vector request
  - A user application can request a state vector that is not available from any loaded SPK file, as long as the state is *derivable* from loaded SPK data
  - A requested state vector may be derived by arithmetically combining ephemeris data from multiple segments belonging to multiple SPK files
    - » Example: state of moon relative to earth = state of moon relative to earth-moon barycenter minus state of earth relative to earth-moon barycenter
  - If reference frame transformations are required to enable addition and subtraction of state vectors, the SPK subsystem will automatically do these transformations as well.
    - » Additional kernels may need to be loaded to support frame transformations: PCK, CK, FK
    - » The "FK" and "Using Frames" tutorials discuss this topic in detail
- See SPK tutorial backup slides for examples
  - Slides are titled "Retrieving State Vectors: Under the Hood"

A Simple Example of Reading an SPK File

Navigation and Ancillary Information Facility

Initialization…typically done once per program execution

```
Tell your program which SPICE files to use ("loading" files)
  CALL FURNSH  ('spk_file_name')
  CALL FURNSH  ('leapseconds_file_name')
```

Better yet, replace these two calls with a single call to a "furnsh kernel" containing the names of all kernel files to load.

Loop... do as many times as you need

```
Convert UTC time to ephemeris time (TDB), if needed
  CALL STR2ET ( 'utc_string', tdb)

Retrieve state vector from the SPK file at your requested time
  CALL SPKEZR (target, tdb, 'frame', 'correction', observer, state, lightime)
                                                    inputs        outputs

Use the returned state vector in computing geometry of interest.
```


Arguments of SPKEZR - 1

Navigation and Ancillary Information Facility

INPUTS

- **TARGET* and OBSERVER*: Character names or NAIF IDs for the end point and origin of the state vector (Cartesian position and velocity vectors) to be returned. Planets, satellites, comets and asteroids all have positive IDs; spacecraft have negative IDs. See the document "NAIF_IDs Required Reading" for a listing of many of these IDs.**
- **TDB: The time at the observer at which the state vector is to be computed. The time system used is Ephemeris Time (ET), now generally called Barycentric Dynamical Time (TDB). The units are number of seconds past the epoch 2000 JAN 01 12:00:00 TDB. SPICE software is available to compute ET based on UTC (SCET) or Spacecraft Clock (SCLK) time.**
- **FRAME: The SPICE name for the reference frame in which your output state vector is to be given. Often 'J2000', but other choices–both inertial and non-inertial–are available. See the documents "NAIF_IDs Required Reading" and "Frames R.R." for details. SPK software will automatically convert data to the frame you specify (if needed).**

* Character names work for the target and observer inputs only if built into SPICE or if registered using the SPICE ID-body name mapping facility. Otherwise use the SPICE numeric ID in quotes, as a character string.

Arguments of SPKEZR - 2

- **CORRECTION: Specification of what kind of correction(s) to apply in computing the output state vector. See the header for subroutine SPKEZR, or the document SPK Required Reading, for details.**
  - **Note: tropospheric and gravitational effects are not included**

OUTPUTS

- **STATE: This is the Cartesian state vector you requested, including corrections. Contains 6 components: three for position (x,y,z) and three for velocity (dx, dy, dz) of the target with respect to the observer.**

- **LIGHTTIME: The one-way light time between the (optionally aberration-corrected) position of target and the geometric position of the observer at the specified epoch.**

A Simple Example of Reading an SPK File

Initialization - typically do this just once per program execution

```
CALL FURNSH ( 'NAIF0007.TLS' )
CALL FURNSH ( 'HUYGENS_3_MERGE.BSP' )
```

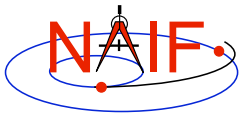
Better to use a "furnsh kernel" instead of these individual FURNSH statements

Repeat in a loop as needed to solve your particular problem

```
CALL STR2ET ('2004 NOV 21 02:40:21.3', TDB )
CALL SPKEZR ('TITAN', TDB, 'J2000', 'LT+S', 'HUYGENS PROBE',
             STATE,  LT )
(Insert additional code here to make derived computations such as
  spacecraft sub-latitude and longitude, lighting angles, etc. Use
  more SPICE subroutines to help.)
```

In this example we get the state (STATE) of Titan as seen from the Huygens probe at the UTC epoch 2004 NOV 21 02:40:21.3. The state vector is given in the J2000 (~ICRF) inertial reference frame and has been corrected for both light time and stellar aberration (LT+S). The one-way light time (LT) is also returned.

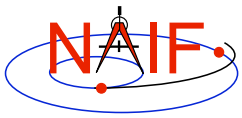
A SPICE leapseconds file (NAIF0007.TLS) is used, as is a SPICE ephemeris file (HUYGENS_3_MERGE.BSP) containing ephemeris data for the Huygens probe (-150), Saturn barycenter (6), Saturn mass center (699), Saturn's satellites (6xx) and the sun (10), relative to the solar system barycenter.



A More Complex Example: Kernel Data Needed

Navigation and Ancillary Information Facility

- **To get states referenced to a non-inertial reference frame, or when the data are provided in a non-inertial frame, typically more kernels will be needed. Examples:**
  - **To get the state of an object relative to a planet in the planet's IAU body-fixed reference frame you'll need:**
    - » **PcK file containing orientation data for the planet**
    - » **SPK(s) for the object, planet, and (typically) planet barycenter**
    - » **LSK**
  - **To get the state of an object in a spacecraft-fixed reference frame you'll need:**
    - » **FK, CK and SCLK for the spacecraft**
    - » **SPK(s) for the spacecraft and object**
    - » **LSK**



A More Complex Case: Reading an SPK File

Navigation and Ancillary Information Facility

Initialization...typically once per program execution

Tell your program which SPICE files to use ("loading" files)

```
CALL FURNSH  ('spk_file_name')
CALL FURNSH  ('leapseconds_file_name')
CALL FURNSH  ('pck_file_name')
```

Better to use a "furnsh kernel" instead of these three individual FURNSH statements

Loop... do as many times as you need

Convert UTC time to ephemeris time (TDB), if needed

```
CALL STR2ET ( 'utc_string', tdb)
```

Get state vector from SPK file at requested time, in planet's IAU body-fixed frame

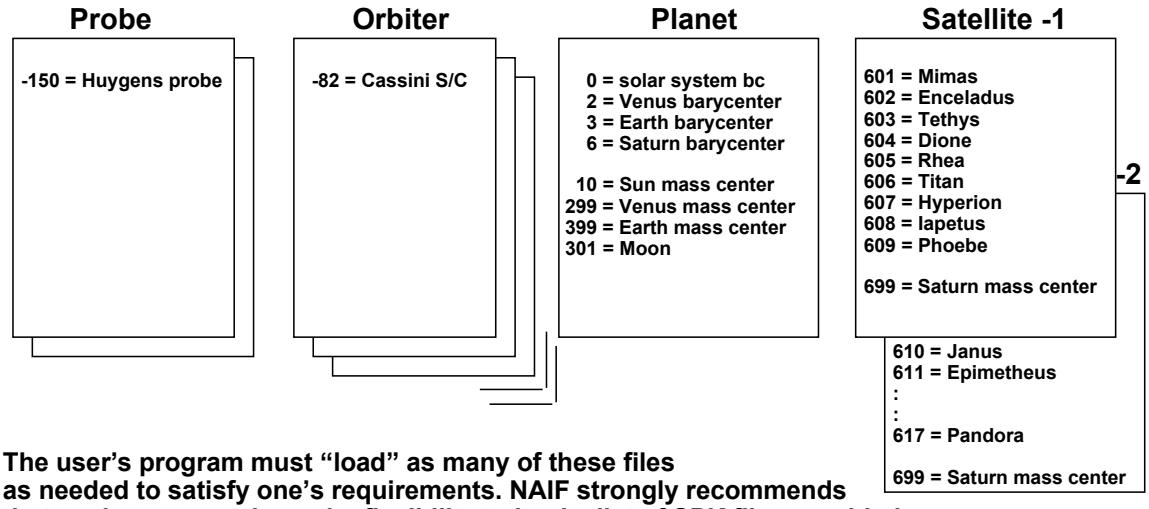
```
CALL SPKEZR (target, tdb, 'IAU_<body_name>', 'correction',
  observer, state, lightime)


(Insert additional code here to make derived computations such as
  spacecraft sub-latitude and longitude, lighting angles, etc. Use
  more SPICE subroutines to help.)
```

Example of Project SPK Files

Navigation and Ancillary Information Facility

This example shows four families of SPK files

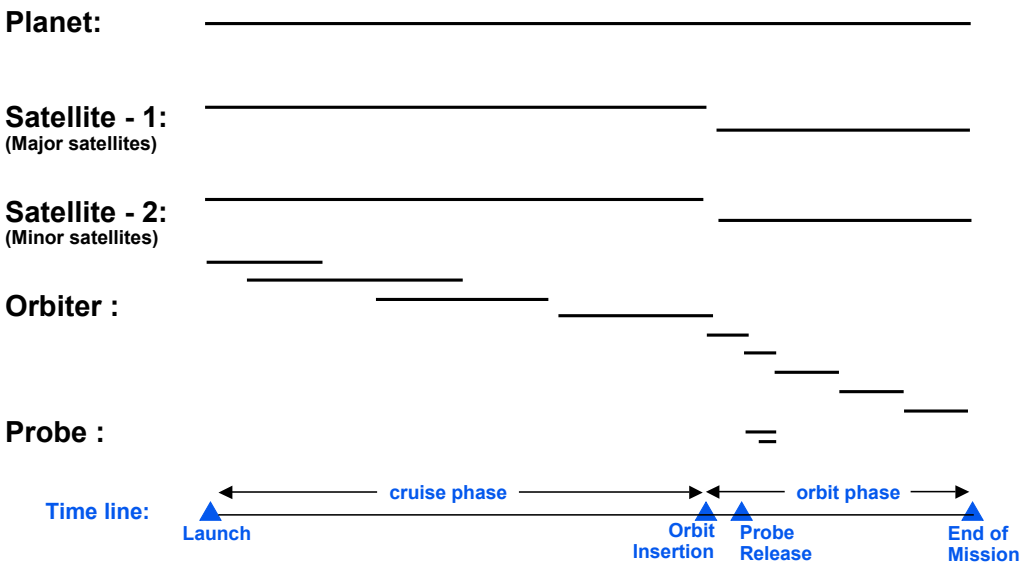


The user's program must "load" as many of these files as needed to satisfy one's requirements. NAIF strongly recommends that such programs have the flexibility to load a list of SPK files provided to the program at run time; this is easily accomplished by listing the SPK files in a "furnsh kernel" using the Toolkit's FURNSH routine.

Possible* SPK File Time Coverages for the Previous Example

Navigation and Ancillary Information Facility

Each bar represents a separate file



* Note: This is likely not a real Cassini scenario; it is simply an illustration of some of the possibilities for ephemeris delivery on a planetary mission.

Four Examples of Generic SPK File Contents

Navigation and Ancillary Information Facility

de405s.bsp
Planetary Ephemeris

0	S.S. BC
1	Merc. BC
199	Mercury
2	Venus BC
299	Venus
3	Earth BC
301	Moon
399	Earth
4	Mars BC
499	Mars
5	Jupiter BC
6	Saturn BC
7	Uranus BC
8	Neptune BC
9	Pluto BC
10	Sun

my_asteroids.bsp
Asteroid Ephemeris

2000253	Mathilde
2000433	Eros

sat081-4.bsp
Satellite Ephemeris

610	Janus
611	Epimetheus
612	Helene
•	
•	
•	
617	Pandora
699	Saturn

jup120_de405.bsp
Merged Ephemeris

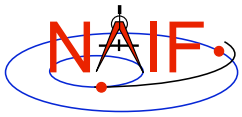
3	Earth BC
399	Earth
5	Jupiter BC
501	Io
502	Europa
503	Ganymede
504	Callisto
599	Jupiter
10	Sun

These kinds of generic SPK files can be obtained from NAIF at any time.

Getting Topocentric States

Navigation and Ancillary Information Facility

- One can make an SPK file containing positions of tracking stations, observatories, roving vehicles, etc.
- This functionality uses the non-inertial frames capability
  - Requires use of a SPICE PcK file to rotate vectors to an inertial frame such as J2000
    - » Can use a low precision body rotation model (IAU)
      - Data are in standard SPICE text PcK files
    - » For earth can use a high precision rotation model ("EOP")
      - Data are in a SPICE binary PcK file
    - » In both cases, you'll also need a size/shape model such as the triaxial ellipsoid models found in standard text PcK files
- Having acquired such an SPK file the user reads this file the same as for any other SPK file
  - Use the NAIF ID of the antenna, observatory or rover as the "target" or "observer" in an SPK reader argument list

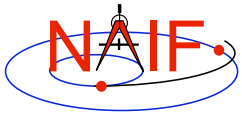


Manipulating and Using SPK Files

Navigation and Ancillary Information Facility

- The producer should provide descriptive information inside an SPK file, in the “comment area”
  - The comments should say when/why/how and for what purpose the file was made
- You can port SPK files between any two computers
- You can subset an SPK file, or merge two or more files
  - The merge may key off of objects, or time, or both
- You can read data from just one, or many* SPK files in your application program
- Don’t forget the precedence rule: data in a later loaded file take precedence over data from an earlier loaded file

(* The allowed number of simultaneously loaded DAF-based files is currently set to 1000.)



Summarizing an SPK File - 1

Navigation and Ancillary Information Facility

- A brief summary can be made using the SPICE Toolkit utility “brief”
  - Summary is for objects present and their start and stop epochs
- At your command line prompt, type the program name (with path), followed by the name of the binary SPK file that is to be summarized
- See the brief User’s Guide or on-line help (%brief -h) for more details

```
%brief  sat052.bsp

 BRIEF --- Brief SPK Summary Tool, Version 2.x
 Times are given as calendar format ephemeris time, not UTC

 SPK file sat052.bsp in brief:

   Bodies: 601, 602, 603, 604, 605, 606, 607, 608, 609, 699

   Begin Ephemeris              End Ephemeris
   -------------------------    -------------------------
   2003 NOV 17, 00:00:00.000    2014 DEC 19, 00:00:00.000
```

Summarizing an SPK File - 2

Navigation and Ancillary Information Facility

- A detailed summary can be made using the Toolkit utility "SPACIT"
- See the SPACIT User's Guide for details

```
 Summary for SPK file: sat052_nov2003_dec2014.bsp

-------------------------------------------------------------------------------
Segment ID    : SAT052
UTC Start time: 2003 NOV 16 23:58:57.817
UTC Stop time : 2014 DEC 18 23:58:57.816
ET Start time :  1.2229920000000E+08
ET Stop time  :  4.7221920000000E+08
Target Body   : MIMAS (601)
Center Body   : SATURN BARYCENTER (6)
Inertial frame: DE-200 (14)
SPK Data Type : Chebyshev polynomials: position only (2)
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
Segment ID    : SAT052
UTC Start time: 2003 NOV 16 23:58:57.817
UTC Stop time : 2014 DEC 18 23:58:57.816
ET Start time :  1.2229920000000E+08
ET Stop time  :  4.7221920000000E+08
Target Body   : ENCELADUS (602)
Center Body   : SATURN BARYCENTER (6)
Inertial frame: DE-200 (14)
SPK Data Type : Chebyshev polynomials: position only (2)
-------------------------------------------------------------------------------
                                    :
                                    :
```

(This is a partial output; not all data could be displayed on this chart)

Summarizing an SPK File - 3

Navigation and Ancillary Information Facility

- A high-level subroutine interface for obtaining SPK coverage information was introduced in the N0058 version of the SPICE Toolkit.
  - Call SPKOBJ to find the set of objects for which a specified SPK provides data.
    - » INPUT: an SPK file name and initialized SPICE integer "Set" data structure. The set may optionally contain ID codes obtained from previous calls.
    - » OUTPUT: the input set, to which have been added (via set union) the ID codes of objects for which the specified SPK provides coverage.

      ```
      CALL SPKOBJ ( SPK, IDSET )
      ```
  - Call SPKCOV to find the window of times for which a specified SPK file provides coverage for a specified body:
    - » INPUT: an SPK file name, body ID code and initialized SPICE double precision "Window" data structure. The window may optionally contain coverage data from previous calls.
    - » OUTPUT: the input window, to which have been added (via window union) the sequence of start and stop times of segment coverage intervals of the specified SPK, expressed as seconds past J2000 TDB.

      ```
      CALL SPKCOV ( SPK, IDCODE, COVER )
      ```
- See the headers of these routines for example programs.
- Also see the CELLS, SETS and WINDOWS Required Reading for background information on these SPICE data types.

Additional Information on SPK

Navigation and Ancillary Information Facility

- For more information about SPK, look at the following:
  - The STATES cookbook program (source code) and its User's Guide
  - The *Most Useful SPICELIB Subroutines* document
  - The *SPK Required Reading* document
  - Headers of the SPKEZR and FURNSH subroutines
  - The Using Frames tutorial
  - The *BRIEF User's Guide* and the *SPACIT User's Guide*
- Related documents:
  - NAIF_IDS Required Reading
  - Frames Required Reading
  - Time Required Reading
  - Kernel Required Reading
- All code and documents noted above are included in every SPICE Toolkit delivery

Backup

Navigation and Ancillary Information Facility

- Problems Using SPK Files
- ID Code Conventions
- Effect of Aberration Corrections
- Retrieving State Vectors: "Under the Hood"

Problems Using SPK Files - 1

Navigation and Ancillary Information Facility

- The file, or files, you loaded do not contain data for both your target and observer bodies
  - You may have loaded the wrong file, or assumed the file contains data that it doesn't
  - You may not have loaded all the files needed
- The file, or files, you loaded do not cover the time at which you requested a state vector
  - This could occur if you've been given a file coverage summary in calendar ET form and you mistook this for UTC
    (ET = UTC + DELTAET, where DELTAET is about 64 secs)
  - This could occur if you are requesting a light-time corrected state and the SPK files being used do not have data at the time that is one-way light-time away* from your ET epoch of interest
    - » * Earlier, for the receive case; later, for the transmission case
- In the above situations you'll get an error message like the following:

```
SPICE(SPKINSUFFDATA) - -
Insufficient ephemeris data has been loaded to
compute the state of xxx relative to yyy.
```


Problems Using SPK Files - 2

Navigation and Ancillary Information Facility

- You have requested aberration-corrected states but the file, or files, you loaded do not contain sufficient data to relate both your target and observer bodies back to the solar system barycenter.
  - You may not have loaded all the files needed
  - You may have assumed the file contains data that it doesn't

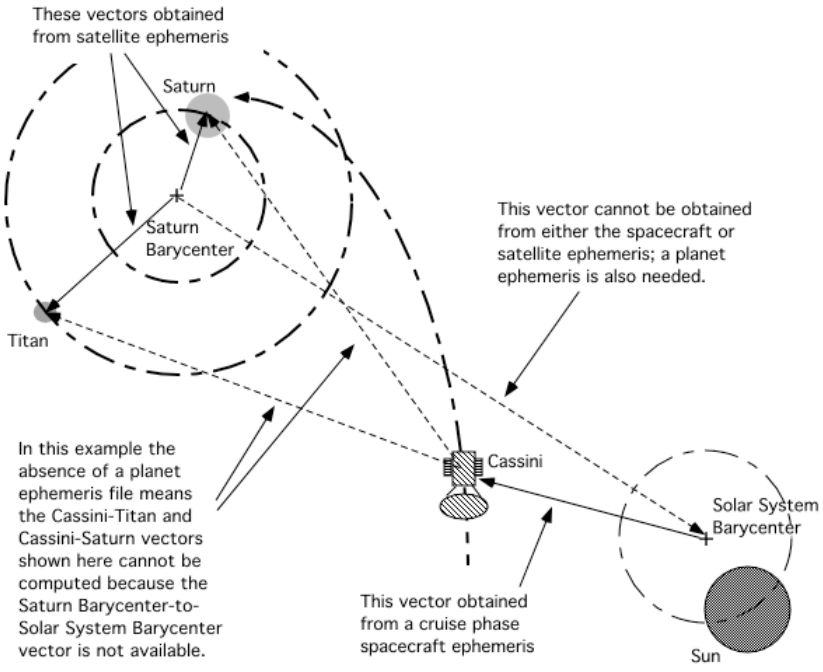
In the above situations you'll get an error message like the following:

```
SPICE(SPKINSUFFDATA) - -
Insufficient ephemeris data has been loaded to
compute the state of xxx relative to yyy.
```

Problems Using SPK Files – 3a

- **An infrequent problem occurs when your SPK file(s) contain data for both target and observer, and cover the period of interest, but ephemeris data for an intermediate body needed to link the target and observer together is missing.**
  - **Example: You load a spacecraft SPK containing ephemeris for Cassini (-82) relative to the solar system barycenter (0), and you load a satellite SPK containing the ephemeris for Titan (606) and Saturn (699) relative to the Saturn barycenter (6). But you forgot to load a planet SPK file that contains data for the Saturn barycenter relative to the solar system barycenter. The SPK software cannot "connect" Cassini to Titan or to Saturn. (See the drawing on the next page.)**
  - **In this case, knowing what is the "Center Body" of movement for each target body is important; this is shown in SPACIT, and in BRIEF summaries if the -c command line option is used.**

Problems Using SPK Files - 3b (drawing)



Problems Using SPK Files - 4

Navigation and Ancillary Information Facility

- **You see an error message to the effect that pole RA (right ascension) data cannot be found**
  - **Probably you are requesting results in a body-fixed frame, but you have not loaded a SPICE PcK file that defines this frame (or, in the case of earth, your binary PcK doesn't cover the epoch of interest to you).**

Problems Using SPK Files - 5

Navigation and Ancillary Information Facility

The problems described here are expected to occur only rarely.

- **You have Toolkit version N0051 or earlier and are trying to read a non-native binary SPK file**
  - **The SPK readers available starting with the version N0052 Toolkit can do on-the-fly translation of non-native binary kernels (except for VAX and Alpha), but this is not the case for earlier Toolkits. (See "porting_kernels.")**
  - **You must get a transfer format SPK file, using ASCII mode of FTP, and convert this to your local binary format using "tobin"**
- **You have an up-to-date SPICE Toolkit, but you attempt to read a non-native binary kernel created using a SPICE Toolkit version N0051 or earlier.**
  - **The SPICE Toolkit assumes the kernel is native if there is no format identifier in the file record.**
  - **The file can be updated on its native system using a post-N0051 SPICE Toolkit by converting it to transfer format using TOXFR and then back to binary format using TOBIN.**
- **You attempt to read a "transfer format" SPK file**
  - **You can read only a binary SPK file with the SPK subroutines; you CAN'T read a "transfer format" file**
    - » **Although not required, binary SPK files often have a name like "xxxxxx.bsp"**
    - » **Although not required, transfer format SPK files often have a name like "xxxxxx.xsp" (formerly "xxxxxx.tsp")**
    - » **Convert a transfer format file to binary format using the "tobin" utility program found in the SPICE Toolkit**
- **You used a transfer format SPK file to move ephemeris information between dissimilar computers, but you get an error when trying to read the newly created binary file on your destination machine**
  - **Probably you failed to use ASCII mode of FTP when transferring the the *.xsp file to your machine**

Problems Using SPK Files - 6

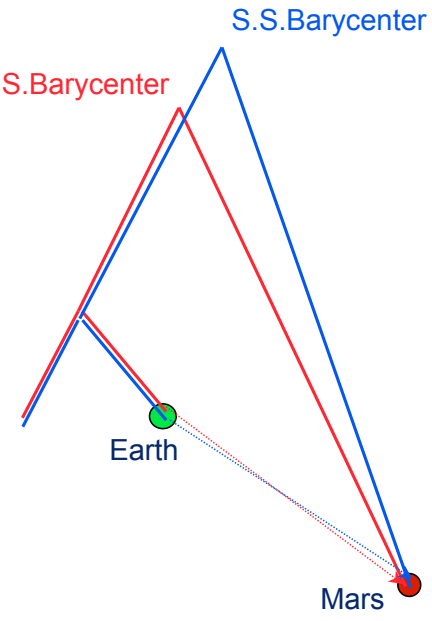
Navigation and Ancillary Information Facility

- **You've loaded sufficient data to "connect" target and observer, but the SPK subsystem can't make the connection.**
  - **This can happen when a high-priority segment that can't be connected to both target and observer "masks" a lower-priority segment that can be connected.**
  - **Example: you want the state of earth as seen from the Galileo orbiter at a specified ephemeris time ET.**
    - » **You have loaded SPK files providing**
      - **The state of the Galileo orbiter relative to the asteroid Gaspra**
      - **The state of the orbiter relative to the sun**
      - **The state of the earth relative to the earth-moon barycenter**
      - **The states of the sun and earth-moon barycenter relative to the solar system barycenter**
    - » **If an SPK segment for the orbiter relative to Gaspra covering ET has higher priority than the segment for the orbiter relative to the sun covering ET, no connection between the orbiter and the earth will be made.**
    - » **Solution:**
      - **Load an SPK file providing the ephemeris of Gaspra relative to the sun or the solar system barycenter (for a time interval containing ET)**
      - **Or: load the SPK file giving the ephemeris of the orbiter relative to Gaspra before the SPK file giving the ephemeris of the orbiter relative to the sun, thereby giving higher priority to the ephemeris of the orbiter relative to the sun.**

Comparing Apples and Oranges

Navigation and Ancillary Information Facility

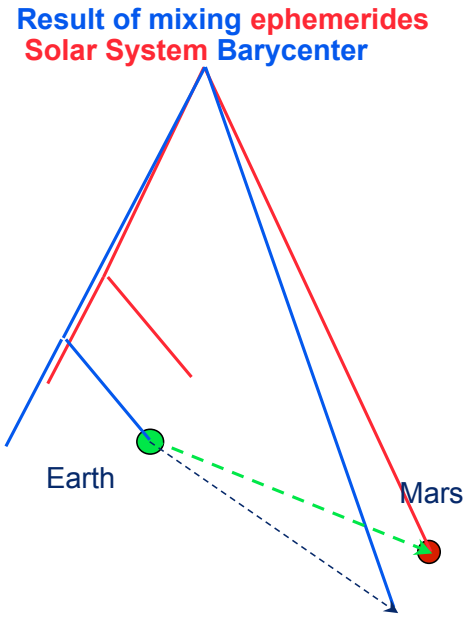
- **With each new integration of the solar system the solar system barycenter moves w.r.t. the planets**
- **Planet to planet offset variations are much smaller than the barycenter to planet variations**



Comparing Apples and Oranges

Navigation and Ancillary Information Facility

- SPICE allows you to "load" different planetary ephemerides (or portions of them)
  - » Potentially can difference positions from different ephemerides to get relative states
- Don't Mix Planetary Ephemerides
- For missions, a consistent set of ephemerides is provided



ID Code Conventions

Navigation and Ancillary Information Facility

- In the SPK subsystem:
  - 1 is equivalent to 199 and 2 is equivalent to 299 because Mercury and Venus haven't any satellites
    - » Objects 199 and 299 are included in standard planet ephemeris files, along with 1 and 2
  - 4 is equivalent to 499 because Mars' satellites have negligible mass as compared to Mars
    - » Object 499 is included in standard planet ephemeris files as well as in Mars satellite ephemeris files
- Caution: the above is NOT the case in PcK files; there you must use only "199", "299" and "499"

Effect of Aberration Corrections - 1

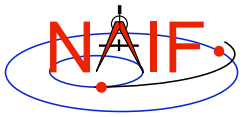
Navigation and Ancillary Information Facility

- **Angular offsets between corrected and uncorrected position vectors over the time span 2004 Jan 1--2005 Jan1**
  - **Mars as seen from MEX:**
    - » **LT+S vs NONE: .0002 to .0008 degrees**
    - » **LT vs NONE: .0006 to .0047 degrees**
  - **Earth as seen from MEX:**
    - » **LT+S vs NONE: .0035 to .0106 degrees**
    - » **LT vs NONE: .0000 to .0057 degrees**
  - **MEX as seen from Earth:**
    - » **LT+S vs NONE: .0035 to .0104 degrees**
    - » **LT vs NONE: .0033 to .0048 degrees**
  - **Sun as seen from Mars:**
    - » **LT+S vs NONE: .0042 to .0047 degrees**
    - » **LT vs NONE: .0000 to .0000 degrees**

Effect of Aberration Corrections - 2

Navigation and Ancillary Information Facility

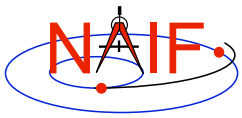
- **Angular offsets between corrected and uncorrected position vectors over the time span 2004 Jan 1--2008 Jan1**
  - **Saturn as seen from CASSINI:**
    - » **LT+S vs NONE: .0000 to .0058 degrees**
    - » **LT vs NONE: .0001 to .0019 degrees**
  - **Titan as seen from CASSINI:**
    - » **LT+S vs NONE: .0000 to .0057 degrees**
    - » **LT vs NONE: .0000 to .0030 degrees**
  - **Earth as seen from CASSINI:**
    - » **LT+S vs NONE: .0000 to .0107 degrees**
    - » **LT vs NONE: .0000 to .0058 degrees**
  - **CASSINI as seen from Earth:**
    - » **LT+S vs NONE: .0000 to .0107 degrees**
    - » **LT vs NONE: .0000 to .0059 degrees**
  - **Sun as seen from CASSINI:**
    - » **LT+S vs NONE: .0000 to .0059 degrees**
    - » **LT vs NONE: .0000 to .0000 degrees**



Retrieving State Vectors: "Under the Hood" - 1

Navigation and Ancillary Information Facility

- **Example: find the geometric state of the MGS orbiter relative to Mars the at observation epoch ET, expressed in the J2000 reference frame.**
  - **CALL SPKEZR ('MGS', ET, 'J2000', 'NONE', 'MARS', STATE, LT)**
  - **The SPK subsystem locates an SPK segment containing the ephemeris of the orbiter relative to Mars covering epoch ET, interpolates the ephemeris data at ET, and returns the interpolated state vector.**
- **Example: find the geometric state of Titan relative to the earth at ET, expressed in the J2000 reference frame.**
  - **CALL SPKEZR ('TITAN', ET, 'J2000', 'NONE', 'EARTH', STATE, LT)**
  - **The SPK subsystem looks up and interpolates ephemeris data to yield:**
    - **The state of the earth relative to the earth-moon barycenter (A)**
    - **The state of the earth-moon barycenter relative to the solar system barycenter (B)**
    - **The state of Titan relative to the Saturn system barycenter at ET (C)**
    - **The state of the Saturn system barycenter relative to the solar system barycenter at ET (D)**
  - **SPKEZR then returns the state vector**
    - **C + D – (A + B)**



Retrieving State Vectors: "Under the Hood" - 2

Navigation and Ancillary Information Facility

- **Example: find the apparent state of the Cassini orbiter relative to the DSN station DSS-14, expressed in the topocentric reference frame centered at DSS-14, at a specified observation epoch ET.**
  - **CALL SPKEZR ('CASSINI', ET, 'DSS-14_TOPO',**
    'LT+S', 'DSS-14', STATE, LT)
  - **The SPK subsystem looks up and interpolates ephemeris data to yield:**
    - **The state of DSS-14 relative to the earth in the ITRF93 terrestrial reference frame (A)**
    - **The state at ET of the earth relative to the earth-moon barycenter in the J2000 reference frame (B)**
    - **The state at ET of the earth-moon barycenter relative to the solar system barycenter in the J2000 frame (C)**
    - **The state at the light time-corrected epoch ET-LT of the Cassini orbiter relative to the Saturn system barycenter (other centers are possible) in the J2000 frame (D)**

- » The state at ET-LT of the Saturn system barycenter relative to the solar system barycenter in the J2000 frame (E)

– The SPK subsystem also looks up transformation matrices to map states:
  - » From the J2000 frame to the ITRF93 terrestrial (earth body-fixed) frame at the observation epoch ET (T1)
  - » From the ITRF93 terrestrial frame to the DSS-14-centered topocentric frame (T2)
– SPKEZR then computes the J2000-relative, light-time corrected observer-target state vector
  - » $E + D - ((T1)^{-1} * A + B + C))$
– SPKEZR corrects this vector for stellar aberration
  - » Call the result "V_J2000_apparent"
– and finally returns the requested state vector in the DSS-14 topocentric reference frame
  - » STATE = T2 * T1 * V_J2000_apparent