



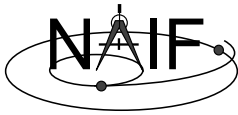
---

Navigation and Ancillary Information Facility

# **Introduction to the SPICE Ephemeris Subsystem SPK**

**Focused on  
reading SPK files**

**October 2007**



---

## **SPK File Contents**

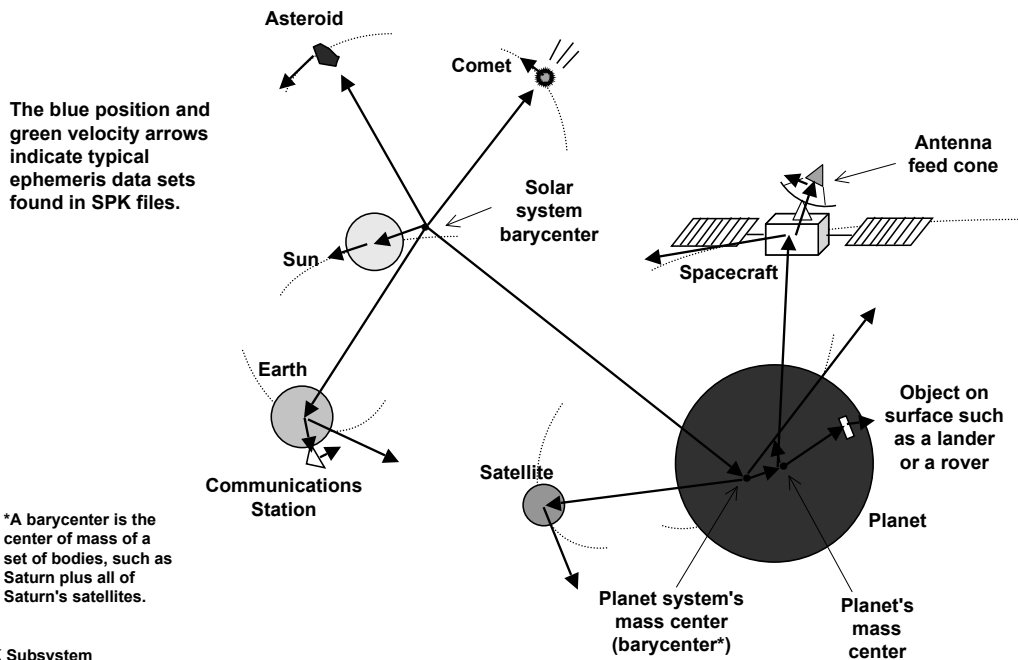
Navigation and Ancillary Information Facility

- **An SPK file contains ephemeris (trajectory) data for "ephemeris objects." We also call ephemeris objects "bodies."**
  - An "Ephemeris object" or "body" can be anything having a trajectory.
  - "Ephemeris data" means the state (position and velocity) of one ephemeris object, the "body," relative to another, the "center," (aka "center of motion") over a given time span, and in a given reference frame.
    - » Position vectors point from the center to the "body".
- **A single SPK file can hold data for one ephemeris object, or for any number of objects.**

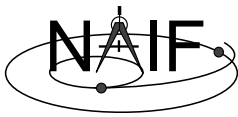


## Examples of Ephemeris Objects

Navigation and Ancillary Information Facility



3



## SPK File Coverage

Navigation and Ancillary Information Facility

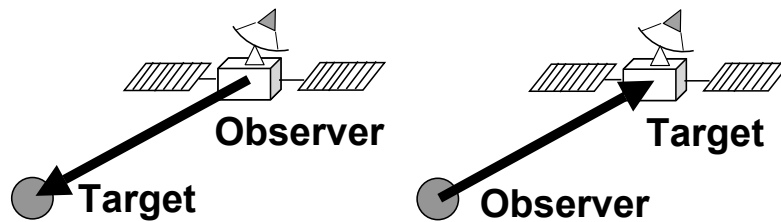
- We call the time period over which an SPK file provides data for an object the “coverage” or “time coverage” for that object.
  - An SPK file’s coverage for an object consists of one or more time intervals.
  - Often, the coverage for all objects in an SPK file is a single, common time interval.
    - » Example: planetary SPK file such as de418.bsp
    - » Counterexample: Cassini tour SPK with merged Huygens probe ephemeris
- For any request time within any time interval comprising the coverage for an object, the SPK system can return a vector representing the state of that object relative to its center.
  - The SPK system will automatically interpolate ephemeris data to produce a state vector at the request time.
  - To a user application, the ephemeris data appear to be continuous over each time interval, even if the stored data are discrete.



## Observers and Targets

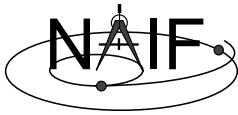
Navigation and Ancillary Information Facility

- The SPK system returns position and state vectors that point from an “observer body” to a “target body.”
  - This terminology reflects typical remote sensing geometry.
- The SPK system allows the user to specify any ephemeris object as the observer, and any object as the target.
  - Provided that the necessary SPICE kernels have been loaded



SPK Subsystem

5



## SPK Works Behind the Scenes

Navigation and Ancillary Information Facility

- The SPK software will automatically chain together the various state vectors needed to connect your "target" to your "observer." (See next chart.)
- SPK software can chain together state vectors provided by multiple SPK files.
- In doing the chaining, the SPK software will also rotate (if needed) the various state vectors into a common reference frame for addition or subtraction, then rotate the result to the reference frame you have specified.

SPK Subsystem

6



## Example of Chaining

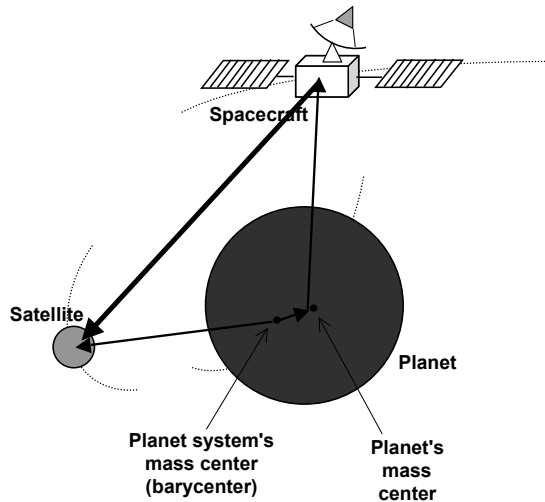
Navigation and Ancillary Information Facility

**Suppose you ask for the position of the satellite relative to the spacecraft.**

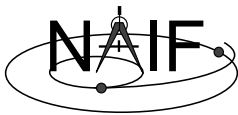
The SPK subsystem will chain together the position vectors indicated by the three blue arrows to give you the position vector indicated by the red arrow: this position connects your target—the satellite—to your observer—the spacecraft.

This would possibly require the loading of two SPK files, one containing data for the spacecraft relative to the planet mass center, and another containing data for the planet mass center and the satellite relative to the planet barycenter.

SPK Subsystem



7



## Retrieving Position or State Vectors - 1

Navigation and Ancillary Information Facility

- To retrieve position, or state (position and velocity) vectors of ephemeris objects from an SPK file one normally needs two kinds of SPICE kernels
  - Leap seconds kernel (LSK)
    - » Used to convert between Coordinated Universal Time (UTC) and Ephemeris Time (ET)
    - » Usually needed since most people work with UTC time
  - Ephemeris kernel(s) (SPK)
    - » Sometimes just one is needed
    - » Sometimes two or more are needed to connect together the "target" and "observer" you have selected.
- Retrieving ephemeris data from an SPK file is also called "reading" the file.
  - This term is frequently used but is not very accurate: the SPK system actually performs interpolation, chains together data from multiple sources, and optionally performs aberration corrections.

SPK Subsystem

8



## Retrieving Position or State Vectors - 2

Navigation and Ancillary Information Facility

Initialization...typically done once per program execution

Tell your program which SPICE files to use (“loading” files)

```
CALL FURNISH ('spk_file_name')
```

```
CALL FURNISH ('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 to

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, light time)
```

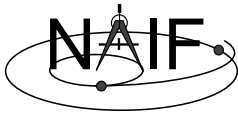
↑  
inputs

↓  
outputs

Use the returned state vector in other SPICE routines to compute observation geometry of interest.

SPK Subsystem

9



## 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.
  - The position component of the requested state vector points from observer to target.
- **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).
- **FRAME:** The SPICE name for the reference frame in which your output state vector is to be given. 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.

SPK Subsystem

10



## Arguments of SPKEZR - 2

Navigation and Ancillary Information Facility

- **CORRECTION:** Specification of what kind of aberration correction(s), if any, to apply in computing the output state vector.
  - Use 'LT+S' to obtain the apparent state of the target as seen by the observer. 'LT+S' invokes light time and stellar aberration corrections.
  - Use 'NONE' to obtain the uncorrected (aka "geometric") state, as given by the source SPK file or files.

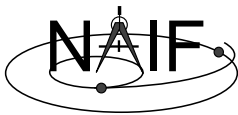
See the header for subroutine SPKEZR, the document SPK Required Reading, or the "Fundamental Concepts" tutorial for details. See the backup charts for examples of aberration correction magnitudes.

### OUTPUTS

- **STATE:** This is the Cartesian state vector you requested. Contains 6 components: three for position (x,y,z) and three for velocity (dx, dy, dz) of the target with respect to the observer. It points from the *observer* to the *target*.
- **LIGHT TIME:** The one-way light time between the (optionally aberration-corrected) position of target and the geometric position of the observer at the specified epoch.

SPK Subsystem

11



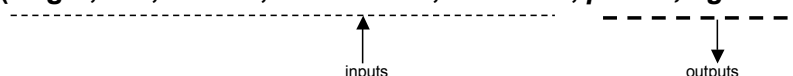
## Retrieving Position Vectors with SPKPOS

Navigation and Ancillary Information Facility

- The SPICE routine SPKPOS is the position-only analog of SPKEZR.
  - The arguments of SPKPOS are identical to those of SPKEZR, except that SPKPOS returns a 3-element position vector instead of a 6-element state vector.
- When velocity data are not needed, SPKPOS has several advantages over SPKEZR:
  - SPKPOS executes more quickly than SPKEZR when stellar aberration corrections are used.
  - SPKPOS can be used when reference frame transformations of velocity are not possible due to lack of C-kernel angular velocity data.

Retrieve a position vector from the SPK file at your requested time:

CALL SPKPOS (target, tdb, 'frame', 'correction', observer, *positrn*, *light time*)



Use the returned position vector in other SPICE routines

SPK Subsystem

12



## A Simple Example of Retrieving State Vectors

Navigation and Ancillary Information Facility

Initialization - typically do this just once per program execution

```
CALL FURNISH ( 'NAIF0008.TLS' )
CALL FURNISH ( 'HUYGENS_3_MERGE.BSP' )
```

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

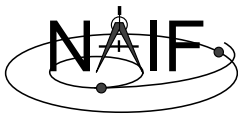
Repeat in a loop if/as needed to solve your particular problem

```
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 returned in the J2000 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.

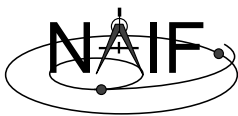
A SPICE leapseconds file (NAIF0008.TLS) is used, as is a SPICE ephemeris file (HUYGENS\_3\_MERGE.BSP) containing ephemeris data for the Huygens probe (-150), Saturn barycenter (6), Saturn mass center (699), Saturn's satellites (6xx) and the sun (10), relative to the solar system barycenter.



## A Slightly More Complex Example - 1 Kernel Data Needed

Navigation and Ancillary Information Facility

- To get states referenced to a non-inertial reference frame, or when the data within the SPK file are provided in a non-inertial frame, typically more kernels will be needed.
  - 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 Slightly More Complex Example - 2 Retrieving State Vectors

Navigation and Ancillary Information Facility

### Obtaining a state in a body-fixed reference frame

Initialization...typically once per program execution

Tell your program which SPICE files to use (“loading” files)

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

} Better to use a “furnsh kernel”  
instead of these three  
individual FURNISH  
statements

Loop... do as many times as you need

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

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

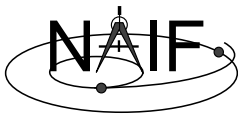
Get state vector from SPK file at requested time, in planet’s IAU body-fixed frame

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

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

SPK Subsystem

15



## SPK File Organization

Navigation and Ancillary Information Facility

- An SPK file is made up of one or more blocks of ephemeris data called “segments.”
  - Each segment contains position and velocity data for one object (the “body”), relative to one observer (the “center”), given in one reference frame, using one mathematical representation (called the SPK “data type”), for a specified time interval.
- An SPK segment, by definition, contains a continuous representation of ephemeris data over an interval of time.
  - You can request a position or state vector at any time within the covered interval.
    - » Achieved through interpolation of discrete data or by evaluating an analytical function, depending on the SPK type.
  - By definition, interpolation across segments or extrapolation beyond the time bounds of a segment is not permitted.

SPK Subsystem

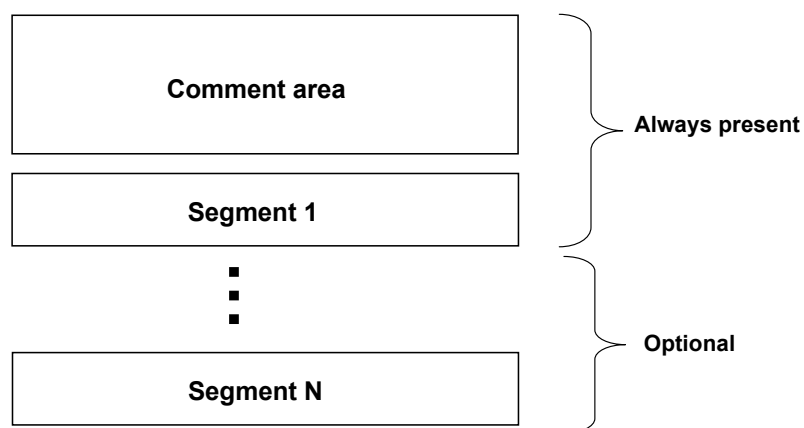
16





## SPK File Structure: Reader's View

Navigation and Ancillary Information Facility



## SPK Segment Order and Priority

Navigation and Ancillary Information Facility

- The segments in an SPK file need not be ordered according to time or body.
- Segment order does imply priority: when two segments from the same SPK file both contain data that satisfy a request, the SPK system selects the segment located later in the file.
  - In the coverage overlap time interval, the lower priority segment is invisible.
- Segments from SPK files loaded later have higher priority: when two segments from two different SPK files both contain data that satisfy a request, the SPK system selects the segment from the SPK file that was loaded later.

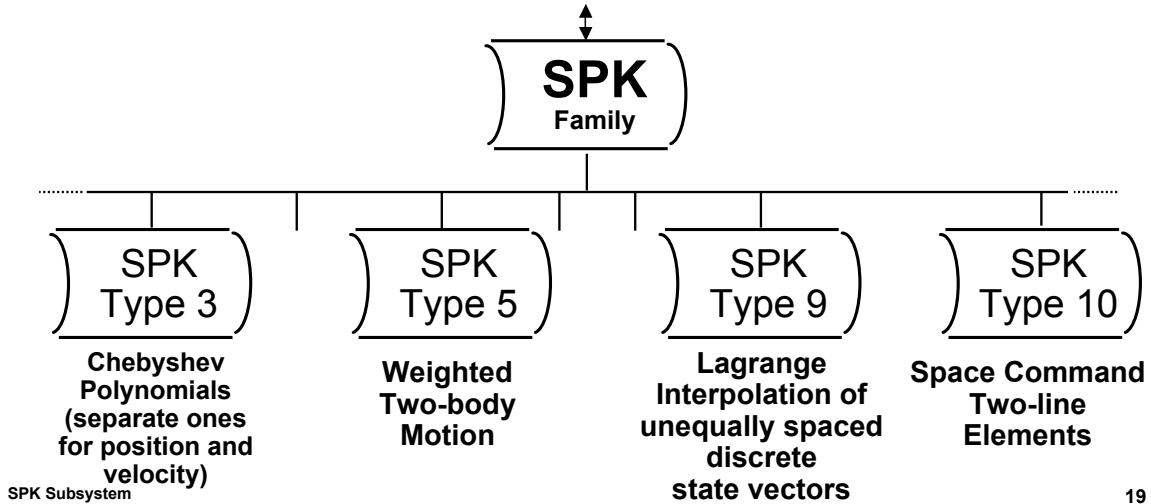


# SPK Data Type Concept

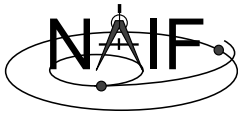
Navigation and Ancillary Information Facility

SPK files contain various mathematical representations of ephemeris data ("data types"), but the high-level user interfaces (SPK "readers") are type-independent:

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



19



## 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.



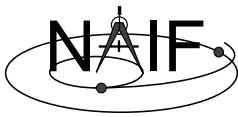
## Widely Used SPK Data Types

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 piecewise 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.
- SPK type 18 (ESOC DDID Hermite/Lagrange interpolation) is used in SPKs for MEX, Rosetta, VEX, and SMART-1.

SPK Subsystem

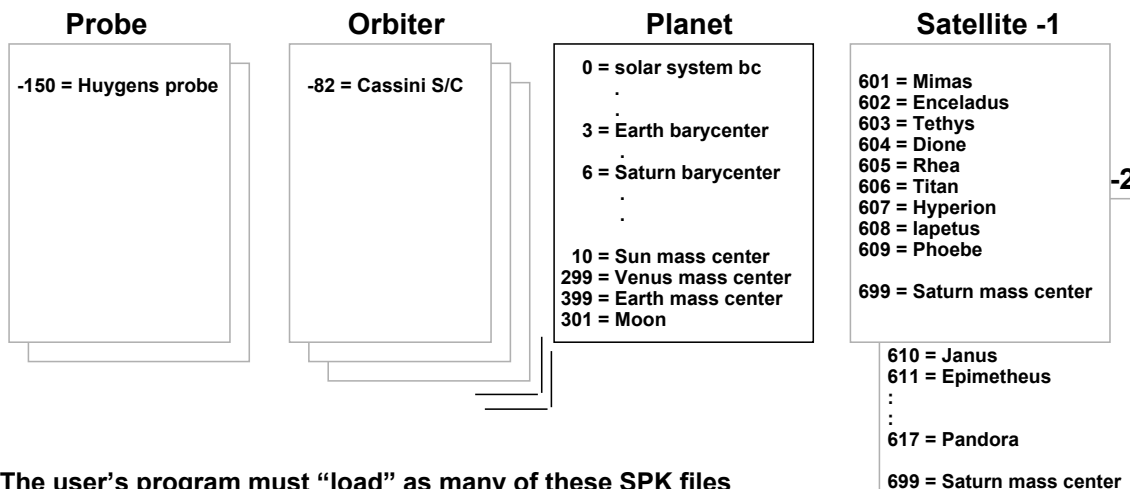
21



## Example of Project SPK Files

Navigation and Ancillary Information Facility

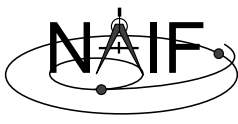
This (made up) example shows four collections of SPK files



The user's program must "load" as many of these SPK files as needed to satisfy his 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 FURNISH routine.

SPK Subsystem

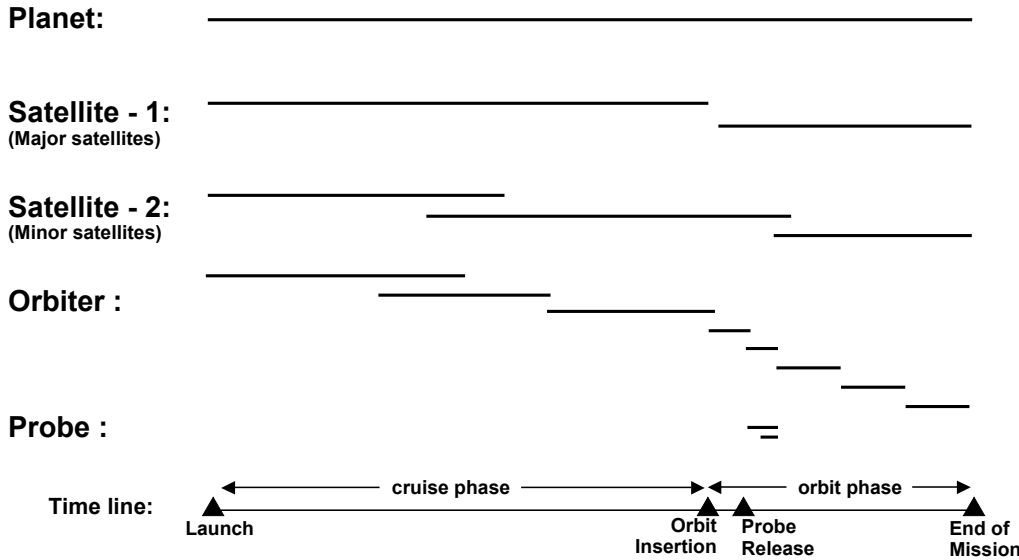
22



# 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.

SPK Subsystem

23



## Planets and Planet Systems

Navigation and Ancillary Information Facility

- Planets and their satellites orbit the planet systems' 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)
- Because Mercury and Venus have no satellites, their 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)

\* These equivalences hold true ONLY in the SPK subsystem, not in the PCK subsystem

SPK Subsystem

24



## 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

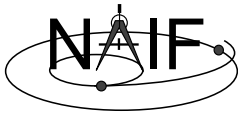
jup120\_de405.bsp  
Merged Ephemeris

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

sat081-4.bsp  
Satellite Ephemeris

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

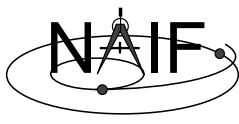
Generic SPK files can be obtained from the NAIF server at any time.



## Body-fixed SPKs

Navigation and Ancillary Information Facility

- One can make an SPK file containing body-fixed positions of tracking stations, observatories, roving vehicles, etc.
- One reads this file the same as for any other SPK file
  - Use the name or NAIF ID of the antenna, observatory or rover as the “target” or “observer” in an SPK reader argument list
  - Also requires use of a SPICE Pck file if you request vectors to be returned in an inertial frame such as J2000

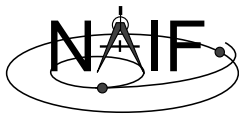


# Manipulating and Using SPK Files

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 070413BP_SCPSE_07097_07121.bsp
Brief. Version: 2.3.1 (SPICE Toolkit N0061)
```

```
Summary for: 070413BP_SCPSE_07097_07121.bsp
```

```
Bodies: CASSINI (-82)      PLUTO BARYCENTER (9)      TETHYS (603)
MERCURY BARYCENTER (1)    SUN (10)                  DIONE (604)
VENUS BARYCENTER (2)      MERCURY (199)             RHEA (605)
EARTH BARYCENTER (3)      VENUS (299)               TITAN (606)
MARS BARYCENTER (4)       MOON (301)                HYPERION (607)
JUPITER BARYCENTER (5)    EARTH (399)               IAPETUS (608)
SATURN BARYCENTER (6)     MARS (499)                PHOEBE (609)
URANUS BARYCENTER (7)     MIMAS (601)               SATURN (699)
NEPTUNE BARYCENTER (8)    ENCELADUS (602)
Start of Interval (ET)    End of Interval (ET)
```

-----  
2007 APR 07 16:22:23.000      2007 MAY 01 09:34:03.000

Note, not UTC!



## 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: sat240.bsp
Leapseconds File   : /kernels/gen/lsk/leapseconds.ker
Summary Type       : Entire File
```

```
-----
Segment ID        : SAT240
Target Body       : Body 601, MIMAS
Center Body       : Body 6, SATURN BARYCENTER
Reference frame    : Frame 1, J2000
SPK Data Type     : Type 3
Description       : Fixed Width, Fixed Order Chebyshev Polynomials: Pos, Vel
UTC Start Time    : 1969 DEC 31 00:00:00.000
UTC Stop Time     : 2019 DEC 02 00:00:00.000
ET Start Time     : 1969 DEC 31 00:00:41.183
ET Stop time      : 2019 DEC 02 00:01:05.183
-----
```

```
-----
Segment ID        : SAT240
Target Body       : Body 602, ENCELADUS
Center Body       : Body 6, SATURN BARYCENTER
Reference frame    : Frame 1, J2000
SPK Data Type     : Type 3
Description       : Fixed Width, Fixed Order Chebyshev Polynomials: Pos, Vel
UTC Start Time    : 1969 DEC 31 00:00:00.000
UTC Stop Time     : 2019 DEC 02 00:00:00.000
ET Start Time     : 1969 DEC 31 00:00:41.183
ET Stop time      : 2019 DEC 02 00:01:05.183
-----
```

⋮

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

SPK Subsystem

29



## Summarizing an SPK File - 3

Navigation and Ancillary Information Facility

- 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.

SPK Subsystem

30

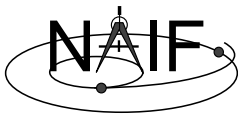


## Additional Information on SPK

---

Navigation and Ancillary Information Facility

- **For more information about SPK, look at the following:**
  - Backup slides in this tutorial
  - 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 FURNISH 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
- Barycenters and Mass Centers
- Effect of Aberration Corrections
- Retrieving State Vectors: "Under the Hood"
- SPK File Structure





## Problems Using SPK Files - 1

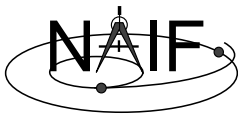
Navigation and Ancillary Information Facility

- 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 65 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.
```

SPK Subsystem

33



## 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.
```

SPK Subsystem

34



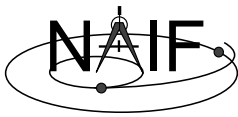


## Problems Using SPK Files - 4

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 \*.xsp file to your machine



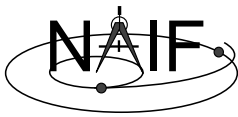
## Problems Using SPK Files - 6

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.

SPK Subsystem

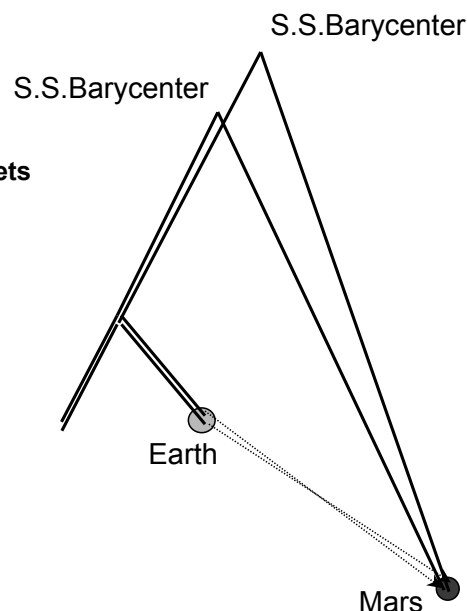
39



## 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



SPK Subsystem

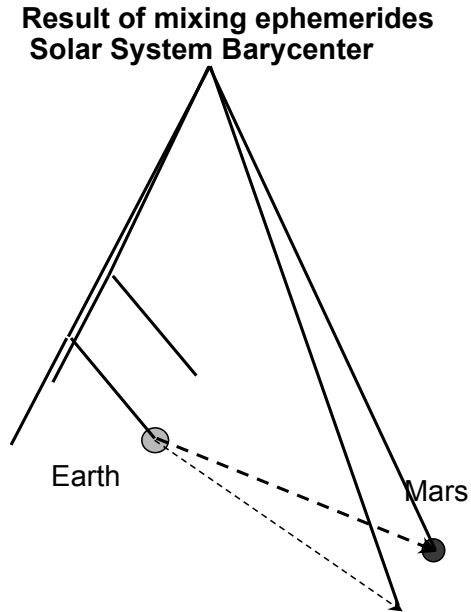
40



# Comparing Apples and Oranges

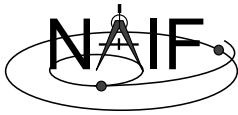
Navigation and Ancillary Information Facility

- **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**



SPK Subsystem

41



## 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”**

SPK Subsystem

42

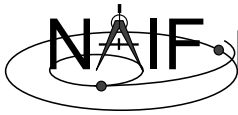


## Barycenters and Mass Centers

Navigation and Ancillary Information Facility

<u>Body Mass Center</u>	<u>System Barycenter</u>	<u>Body center offset from Barycenter (km)*</u>	<u>Offset as % of body radius*</u>
Sun (10)	SSB (0)	1,378,196	198%
Mercury (199)	M. BC (1)	0	0
Venus (299)	V. BC (2)	0	0
Earth (399)	E. BC (3)	4942	77%
Mars (499)	M. BC (4)	~0	~0
Jupiter (599)	J. BC (5)	220	0.3%
Saturn (699)	S. BC (6)	312	0.5%
Uranus (799)	U. BC (7)	43	0.17%
Neptune (899)	N. BC (8)	74	0.3%
Pluto (999)	P. BC (9)	2080	174%

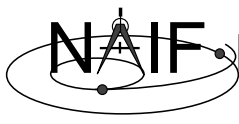
\* Estimated maximum values over the time range 2000-2050



## Effect of Aberration Corrections - 1

Navigation and Ancillary Information Facility

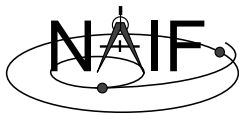
- **Angular offsets between corrected and uncorrected position vectors over the time span 2004 Jan 1--2005 Jan1**
  - Mars as seen from MEX:
    - » LT+S vs NONE: .0002 to .0008 degrees
    - » LT vs NONE: .0006 to .0047 degrees
  - Earth as seen from MEX:
    - » LT+S vs NONE: .0035 to .0106 degrees
    - » 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 Position or State Vectors

Navigation and Ancillary Information Facility

- **The SPICE SPK subsystem looks up ephemeris data as needed to satisfy a user's request**
  - A user application can request a state vector that is not directly available from a loaded SPK file as long as the state is derivable from the segments contained in all loaded SPK files
  - 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 the following SPK tutorial backup slides for examples**
  - Slides are titled "Retrieving State Vectors: Under the Hood"



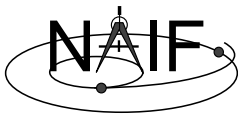
## Retrieving State Vectors: "Under the Hood" - 1

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
    - »  $C + D - (A + B)$

SPK Subsystem

47



## 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)

SPK Subsystem

48





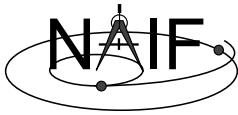
## Retrieving State Vectors: "Under the Hood" - 3

Navigation and Ancillary Information Facility

- » 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
  - »  $STATE = T2 * T1 * V\_J2000\_apparent$

SPK Subsystem

49



## SPK File Structure

Navigation and Ancillary Information Facility

### Example: SPK file with 27 segments

Records are fixed-length: 1024 bytes

ID	WORD	ND	NI	IFNAME	FWD	BWD	FREE	BFF	0 PAD	FTP	0 PAD
Comment area (variable number of records )										U*	
N/P/C	D1	D2	... Descriptor record ...							D25	
I1	I2	... Segment ID ("Name") record ...							I25	U	
Segment 1 (variable number of records )											
Segment 2 (variable number of records )											
⋮											
Segment 25 (variable number of records )										U*	
N/P/C	D26	D27	... Descriptor record ...							U*	
I26	I27	... Segment ID ("Name") record ...							U*	U	
Segment 26 (variable number of records )											
Segment 27 (variable number of records )										U*	

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 Subsystem

Diagram not to scale.

50



# 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**
  - One 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 10000 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