Introduction to the SPICE Ephemeris Subsystem SPK

Focused on reading SPK files

April 2016

First… clear your mind!

- SPK is probably unlike any previous ephemeris or trajectory representation you've used or heard about.
- We think you'll find it to be *much* more capable than other ephemeris system architectures.
  - As such, it's also a bit more complicated to grasp.

- ***Don't panic! Shortly you'll be reading SPK files like a pro.***

Overview of SPICE Ephemeris Data

A Picture is worth …

Navigation and Ancillary Information Facility

- **We’ll start with a mostly pictorial view of ephemeris data and SPK files, just to ease you into this topic.**

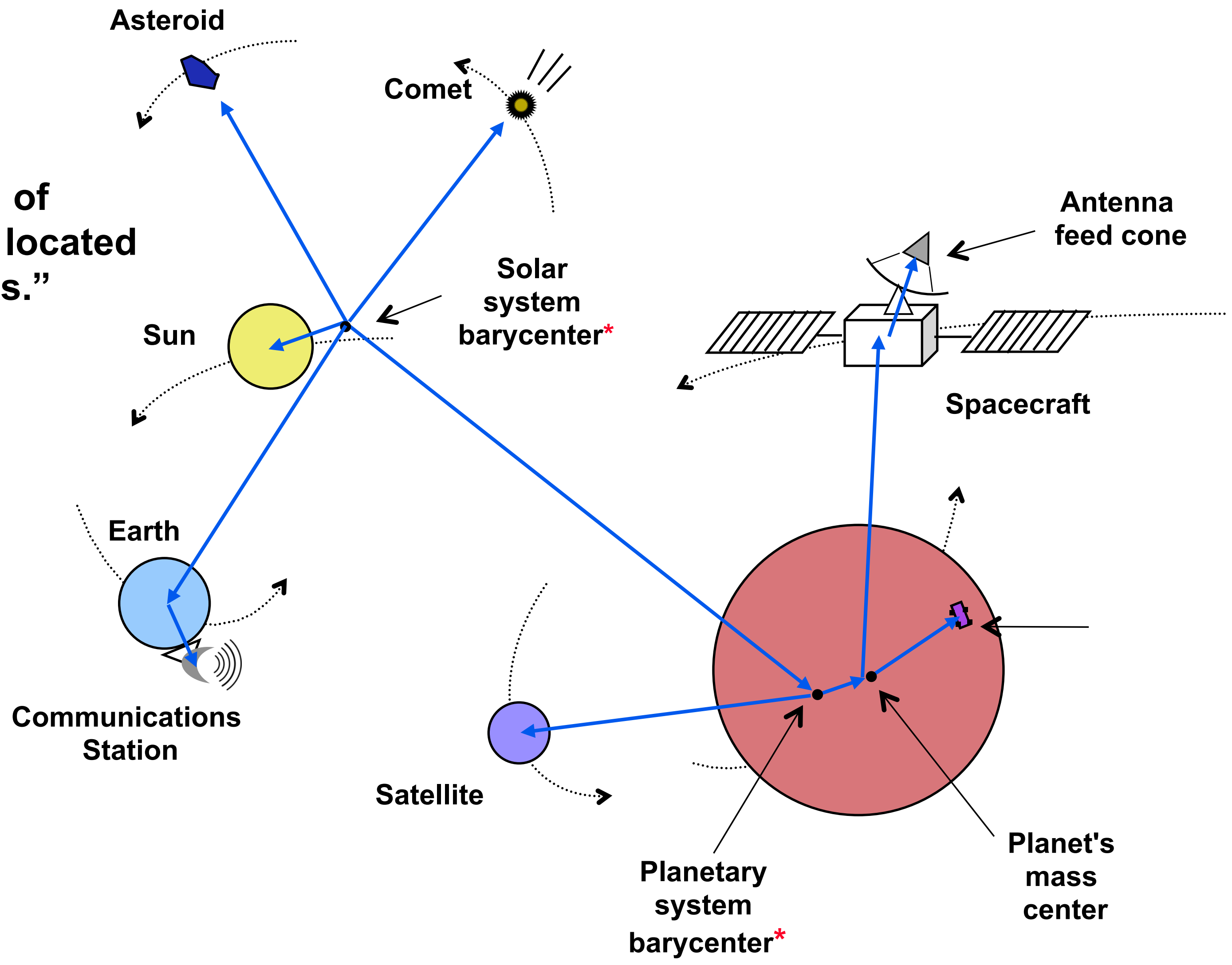
SPICE Ephemeris Data

- An SPK file contains ephemeris (trajectory) data for "ephemeris objects."
  - "Ephemeris" means position and velocity as a function of time
    - » Position + velocity is often referred to as "state"
- "Ephemeris objects" are spacecraft, planets, satellites, comets and asteroids.
  - In SPICE the following are also ephemeris objects:
    - » the center of mass of our solar system (solar system barycenter)
    - » the center of mass of a planet/satellite system (planet barycenter)
    - » a rover on the surface of a body
    - » a camera on top of a mast on a lander
    - » a transmitter cone on a spacecraft
    - » a deep space communications antenna on the earth
- A single SPK file can contain data for multiple ephemeris objects, and often does
- See the next page for a pictorial representation of some of these objects.

Examples of Ephemeris Objects

Navigation and Ancillary Information Facility

The head and the tail of every blue arrow are located at “ephemeris objects.”



*A barycenter is the center of mass of a system of bodies, such as Saturn plus all of Saturn's satellites, or, the sun and all the planets, satellites, comets and asteroids in the solar system.

Imagine Some Ephemeris Data

```
epoch_1,  x1,  y1,  z1,  vx1,  vy1,  vz1
epoch_2,  x2,  y2,  z2,  vx2,  vy2,  vz2
epoch_3,  x3,  y3,  z3,  vx3,  vy3,  vz3
epoch_4,  x4,  y4,  z4,  vx4,  vy4,  vz4
 ………………  etc. ………………
 ………………  etc. ………………
epoch_n,  xn,  yn,  zn,  vxn,  vyn, vzn
```

Perhaps this is an ASCII table or an Excel spreadsheet containing rows of time-tagged Cartesian state vectors

“epoch = “time”

It may not be written inside the table or spreadsheet, but perhaps an interface agreement somehow tells you:

- what object–the “target”–this ephemeris is for
- what is the name of the reference frame (“coordinate frame”) in which the data are given
- what is the center of motion of the target
- what time system is being used for the epochs
- maybe also what are the start and stop times of the file
  - » meaning, what are “epoch_1” and “epoch_n”

Imagine a Simple Ephemeris File

```
epoch_1,  x1,  y1,  z1,  vx1,  vy1,  vz1
epoch_2,  x2,  y2,  z2,  vx2,  vy2,  vz2
epoch_3,  x3,  y3,  z3,  vx3,  vy3,  vz3
epoch_4,  x4,  y4,  z4,  vx4,  vy4,  vz4
 .................  etc. ..................
 .................  etc. ..................
epoch_n,  xn,  yn,  zn,  vxn,  vyn,  vzn
```

We’ll represent that simple ephemeris file as a “block” like this.

Imagine a Simple Ephemeris File

```
epoch_1,  x1,  y1,  z1,  vx1,  vy1,  vz1
epoch_2,  x2,  y2,  z2,  vx2,  vy2,  vz2
epoch_3,  x3,  y3,  z3,  vx3,  vy3,  vz3
epoch_4,  x4,  y4,  z4,  vx4,  vy4,  vz4
 ..................  etc. ...................
 ..................  etc. ...................
epoch_n,  xn,  yn,  zn,  vxn,  vyn,  vzn
```

We’ll represent that simple ephemeris file as a “block” like this.

This becomes the basis of a “segment” in an SPK file.

An SPK “Segment”

Target, Ref Frame ID, Center of Motion, T_{start}, T_{stop}

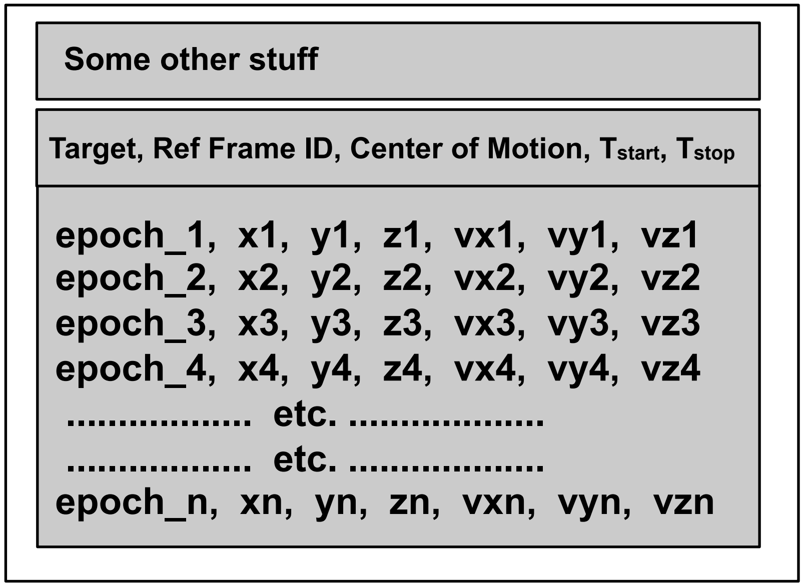
epoch_1, x1, y1, z1, vx1, vy1, vz1
epoch_2, x2, y2, z2, vx2, vy2, vz2
epoch_3, x3, y3, z3, vx3, vy3, vz3
epoch_4, x4, y4, z4, vx4, vy4, vz4
.... etc. ....
.... etc. ....
epoch_n, xn, yn, zn, vxn, vyn, vzn

One segment

We insert this meta-data into the segment

- what is the “target” – the object this ephemeris is for
- what is the “observer” – the center of motion of the target
- what is the ID of the reference frame (“coordinate frame”) in which the data are given
- maybe also what are the start and stop times of the file, T_{start} and T_{stop}
  - meaning, what are “epoch_1” and “epoch_n”

A Simple SPK File



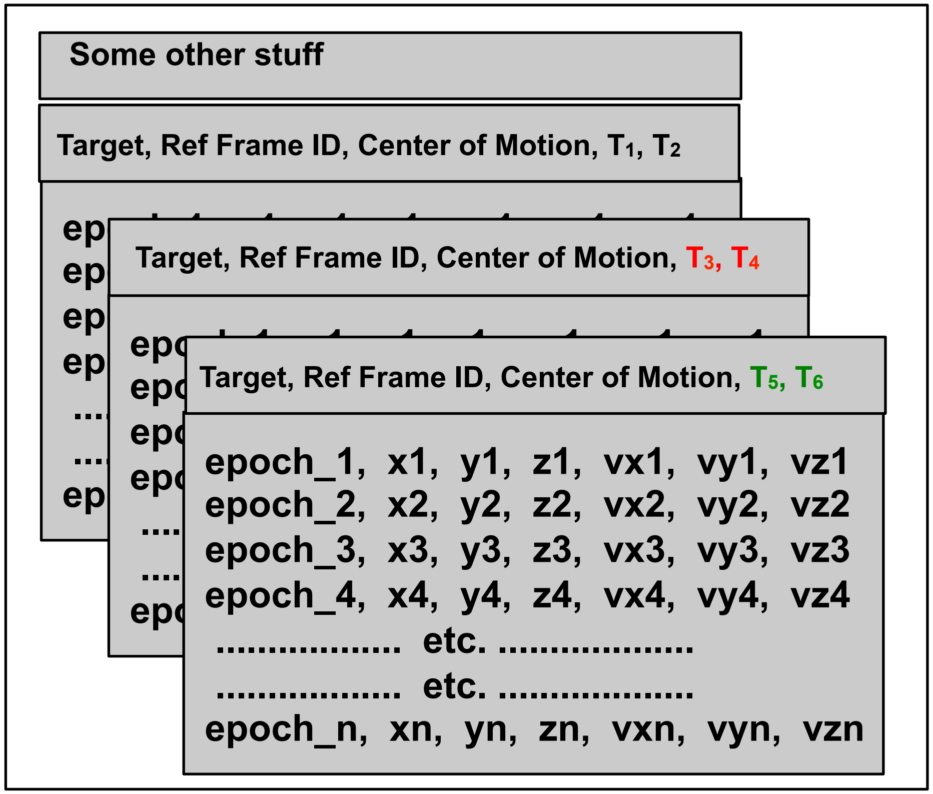
Some other stuff

Target, Ref Frame ID, Center of Motion, T_{start}, T_{stop}

epoch_1, x1, y1, z1, vx1, vy1, vz1
epoch_2, x2, y2, z2, vx2, vy2, vz2
epoch_3, x3, y3, z3, vx3, vy3, vz3
epoch_4, x4, y4, z4, vx4, vy4, vz4
.... etc. ....
.... etc. ....
epoch_n, xn, yn, zn, vxn, vyn, vzn

- This very simple SPK file is made up of a single segment containing ephemeris data:
  - for a single object (perhaps a spacecraft, an asteroid, or …whatever)
  - having a single center of motion
  - given in a single reference frame (“coordinate frame”)
  - with data spanning from T_{start} to T_{stop}

A More Substantial SPK File



Some other stuff

Target, Ref Frame ID, Center of Motion, T_1, T_2

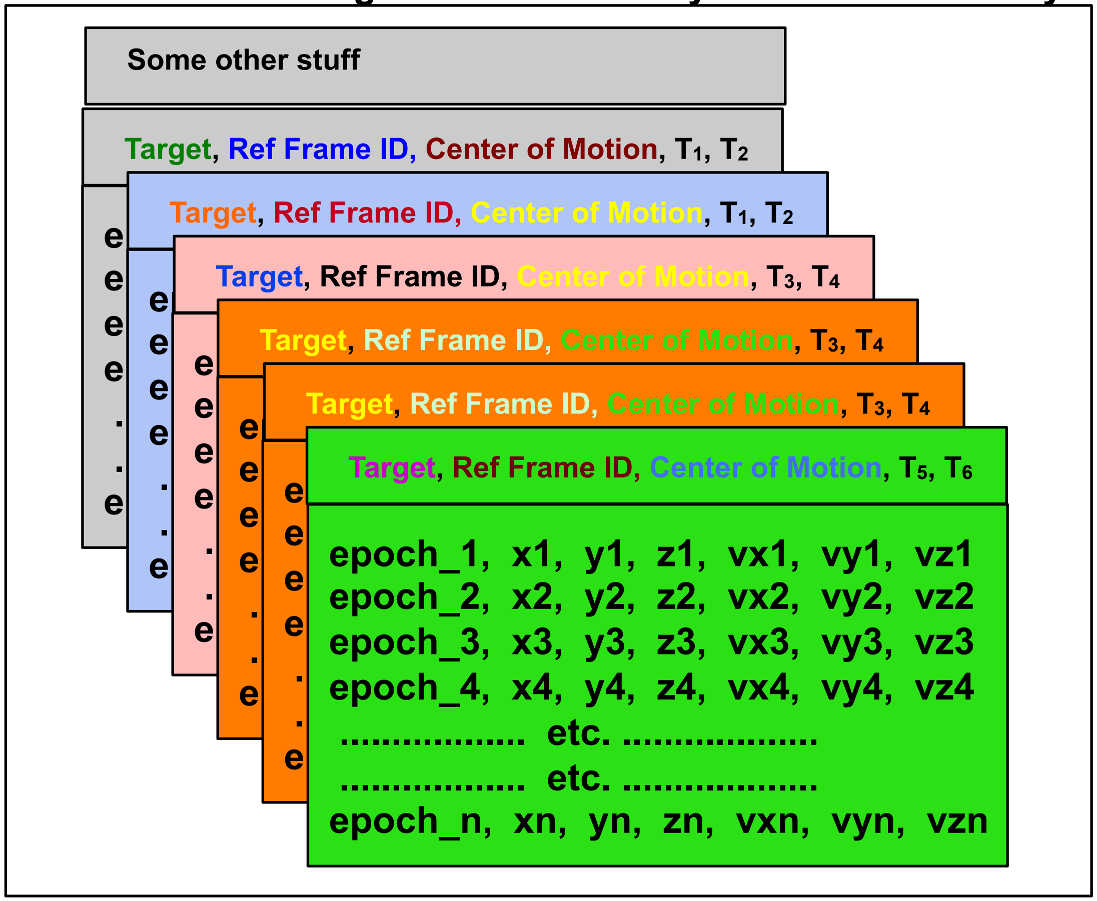
Target, Ref Frame ID, Center of Motion, T_3, T_4

Target, Ref Frame ID, Center of Motion, T_5, T_6

epoch_1, x1, y1, z1, vx1, vy1, vz1
epoch_2, x2, y2, z2, vx2, vy2, vz2
epoch_3, x3, y3, z3, vx3, vy3, vz3
epoch_4, x4, y4, z4, vx4, vy4, vz4
.... etc. ....
.... etc. ....
epoch_n, xn, yn, zn, vxn, vyn, vzn

- This more substantial SPK is made up of multiple segments containing ephemeris data:
  - for a single object (perhaps a spacecraft, an asteroid, or …???)
  - having a single center of motion
  - given in a single reference frame (“coordinate frame”)
  - with data spanning from T_1 to T_6

An Even More Substantial SPK File



Some other stuff

Target, Ref Frame ID, Center of Motion, T_1, T_2

Target, Ref Frame ID, Center of Motion, T_1, T_2

Target, Ref Frame ID, Center of Motion, T_3, T_4

Target, Ref Frame ID, Center of Motion, T_3, T_4

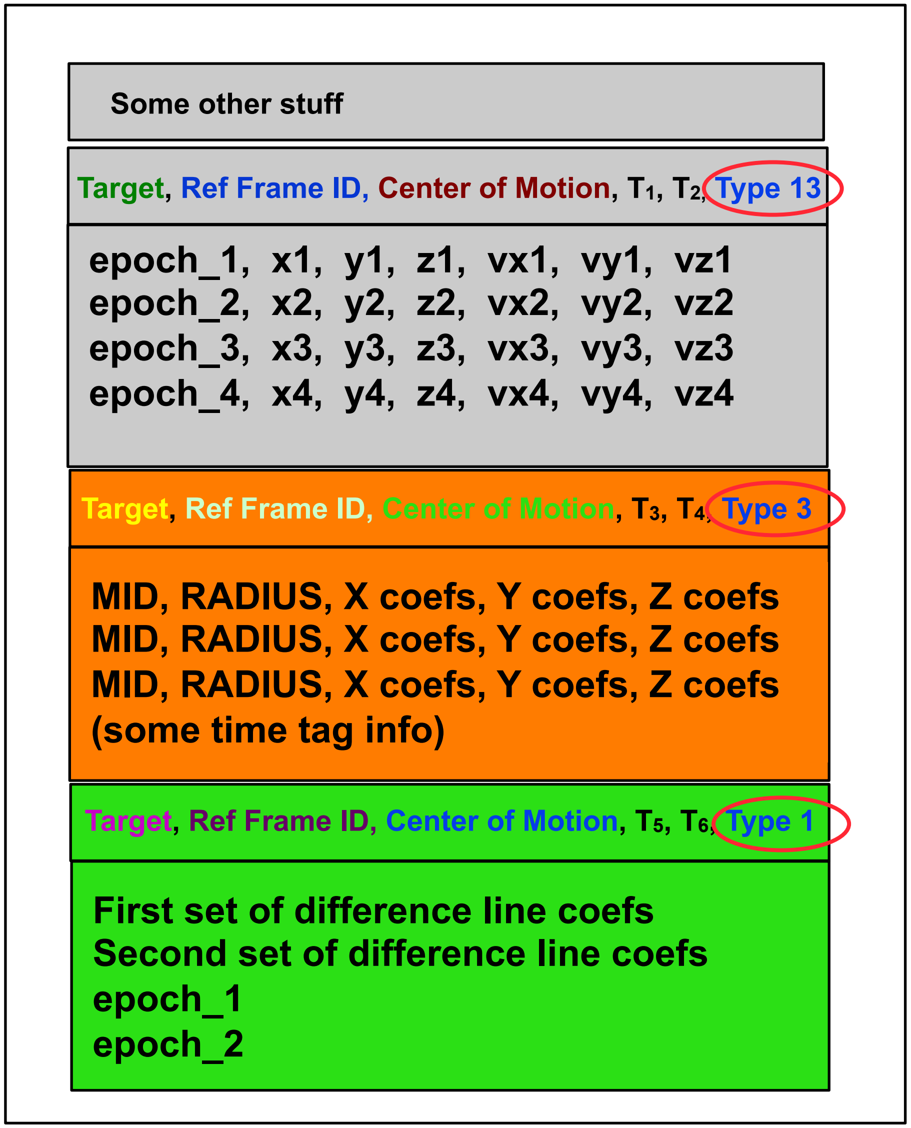
Target, Ref Frame ID, Center of Motion, T_3, T_4

Target, Ref Frame ID, Center of Motion, T_5, T_6

```
epoch_1,  x1,  y1,  z1,  vx1,  vy1,  vz1
epoch_2,  x2,  y2,  z2,  vx2,  vy2,  vz2
epoch_3,  x3,  y3,  z3,  vx3,  vy3,  vz3
epoch_4,  x4,  y4,  z4,  vx4,  vy4,  vz4
 .................  etc. ..................
 .................  etc. ..................
epoch_n,  xn,  yn,  zn,  vxn,  vyn,  vzn
```

- **This even more substantial SPK contains multiple segments having:**
  - different objects
  - different centers of motion
  - different reference frames
  - different pairs of start and stop times

SPK “Type” Info in each Segment



Some other stuff

Target, Ref Frame ID, Center of Motion, T_1, T_2, Type 13

epoch_1, x1, y1, z1, vx1, vy1, vz1
epoch_2, x2, y2, z2, vx2, vy2, vz2
epoch_3, x3, y3, z3, vx3, vy3, vz3
epoch_4, x4, y4, z4, vx4, vy4, vz4

Target, Ref Frame ID, Center of Motion, T_3, T_4, Type 3

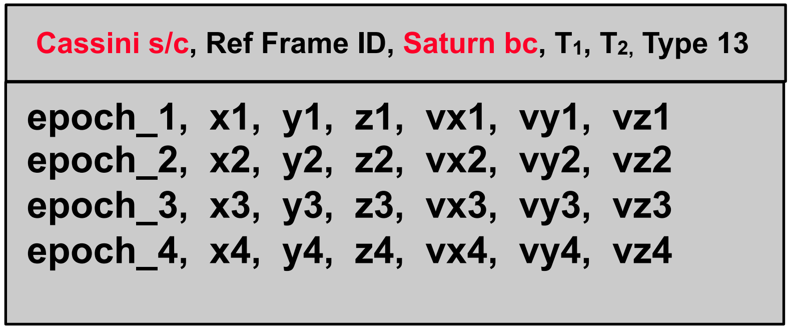
MID, RADIUS, X coefs, Y coefs, Z coefs
MID, RADIUS, X coefs, Y coefs, Z coefs
MID, RADIUS, X coefs, Y coefs, Z coefs
(some time tag info)

Target, Ref Frame ID, Center of Motion, T_5, T_6, Type 1

First set of difference line coefs
Second set of difference line coefs
epoch_1
epoch_2

- Each segment can contain a different type of ephemeris data (as long as it’s been built into the SPK subsystem). Examples:
  - Discrete state vectors
  - Chebyshev polynomials
  - Difference lines (unique to JPL)
  - Etc., etc.
- Each segment has the SPK Type stored in its meta-data record.
- Toolkit software knows how to evaluate each Type – no worries for you!

SPK Data are Continuous



Cassini s/c, Ref Frame ID, Saturn bc, T_1, T_2, Type 13

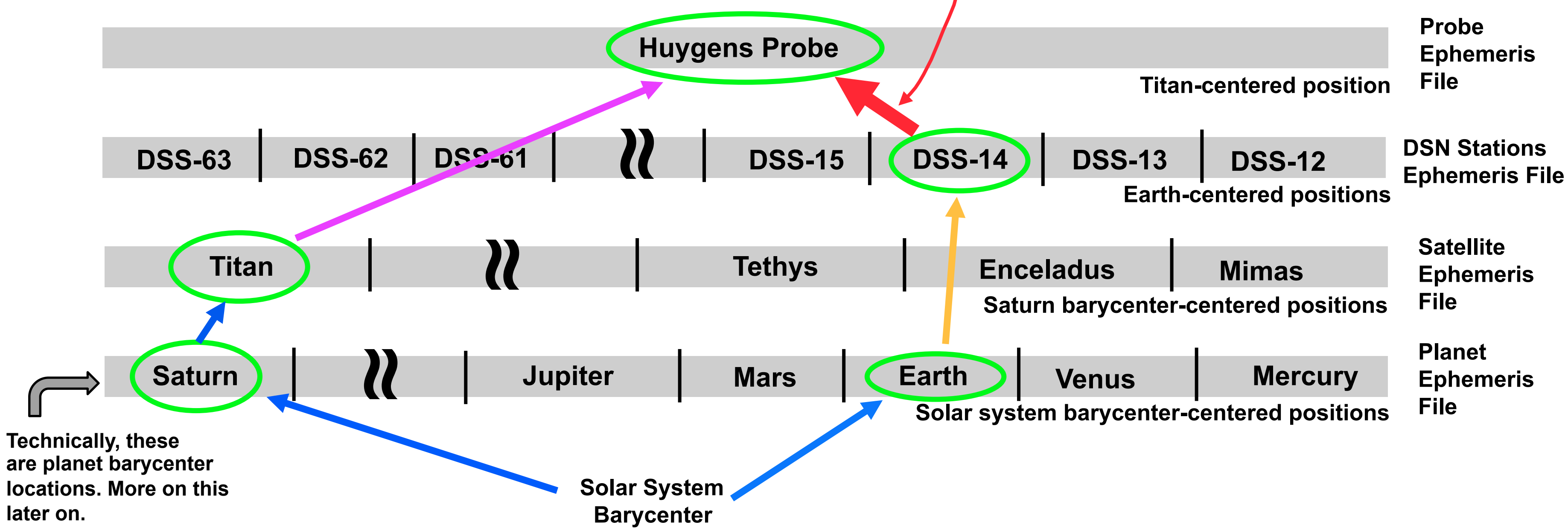
```
epoch_1,  x1,  y1,  z1,  vx1,  vy1,  vz1
epoch_2,  x2,  y2,  z2,  vx2,  vy2,  vz2
epoch_3,  x3,  y3,  z3,  vx3,  vy3,  vz3
epoch_4,  x4,  y4,  z4,  vx4,  vy4,  vz4
```

- **Within the time bounds (T_1, T_2) of a segment, SPICE software will return a result–a state vector consisting of position and velocity–at any epoch… not just at the epochs of the ephemeris records**

- **In the example above, SPICE will return the position and velocity (the state) of the Cassini spacecraft relative to the Saturn barycenter at any time *t* where: $T_1 \leq t \leq T_2$**

SPICE Chains SPK Data

- SPICE automatically searches across all loaded SPK files to find the segments needed to compute the vectors needed to obtain the result the customer has asked for. SPICE chains these together using addition and subtraction.
  - In this example the user wants the position of the Huygens probe sitting on the surface of Titan as seen from Deep Space Station 14.
  - SPICE computes this by chaining the gold, blue and violet chunks.



Technically, these are planet barycenter locations. More on this later on.

Automated Frame Transformation

- **As part of the “chaining” process just mentioned…**
  - **position vectors are automatically rotated into a consistent reference frame**
  - **the final vector is rotated into the output reference frame requested by the user**

Reference Frames Used

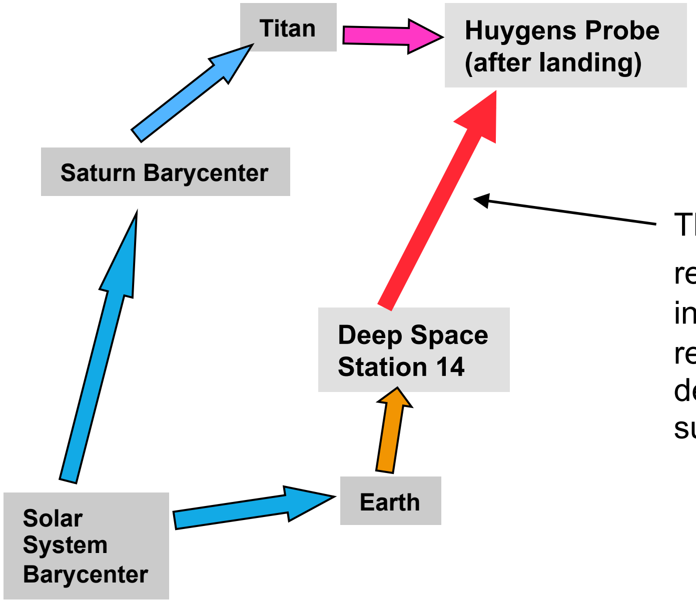
🟦 International Celestial Reference Frame (J2000)

🟪 Titan body-fixed frame (IAU_TITAN)

🟧 International Terrestrial Reference Frame (ITRF93)

🟥 DSS-14 topocentric reference frame (DSS-14_TOPO)

Ephemeris Segments Used



The position of the Huygens Probe relative to DSS-14, at time *t*, given in the DSS-14 topocentric reference frame, can be determined using a SINGLE Toolkit subroutine call.

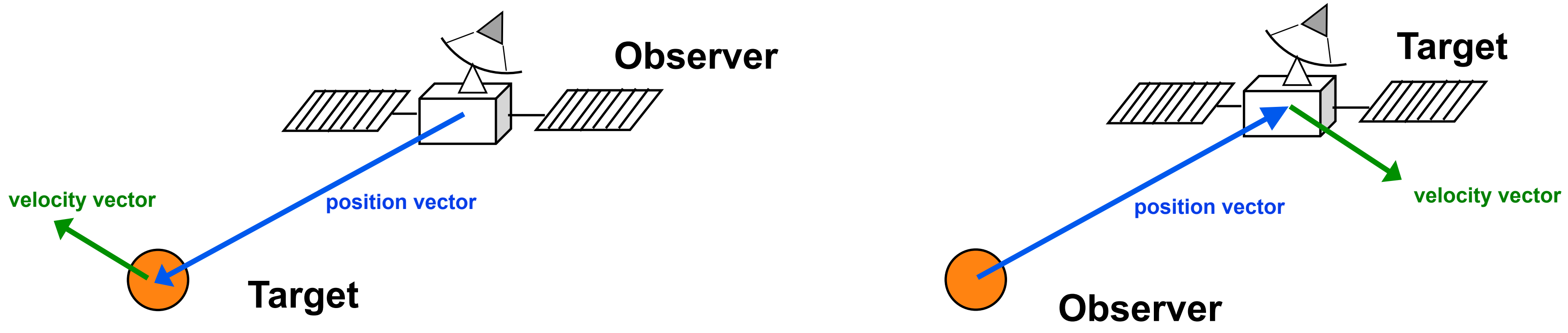
A single subroutine call does it all!

```
CALL SPKPOS (‘HUYGENS_PROBE’, t, ‘DSS-14_TOPO’, LT+S, ‘DSS-14’, POSITION, LT)
```

Now, some details

Reading an SPK: Observers and Targets

- When you read an SPK file you specify which ephemeris object is to be the “target” and which is to be the “observer.”
- The SPK system returns the state of the target relative to the observer.
  - The position data point from the “observer” to the “target.”
  - The velocity is that of the “target” relative to the “observer.”

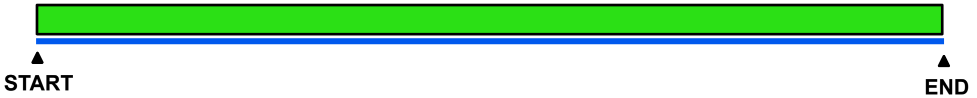


- Any ephemeris object can be a target or an observer

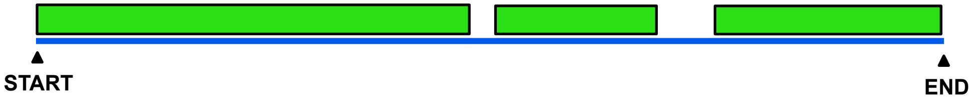
SPK File Coverage - 1

- **The time period over which an SPK file provides data for an ephemeris object is called 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.**

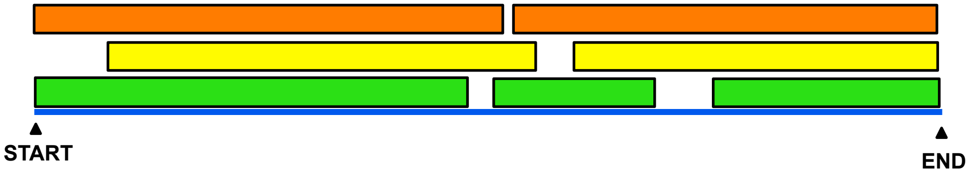
SPK containing data for one object with no data gaps



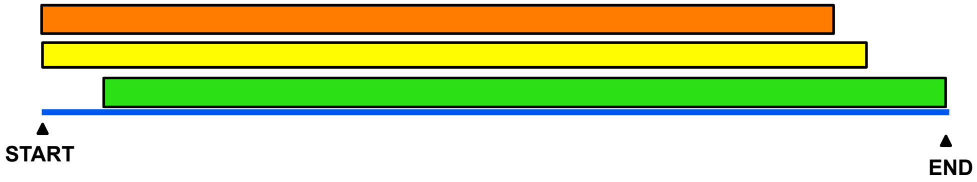
SPK containing data for one object, with two data gaps



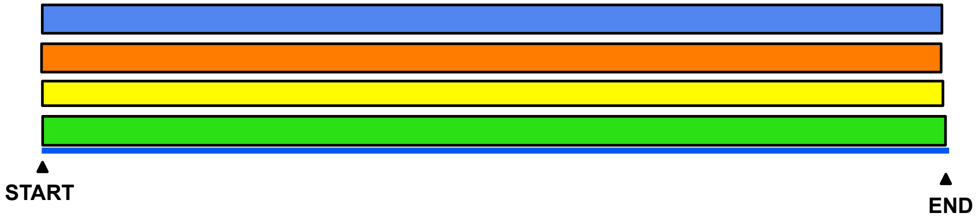
SPK containing data for three objects, each having different data gaps



SPK containing data for three objects, each having different coverage but with no data gaps



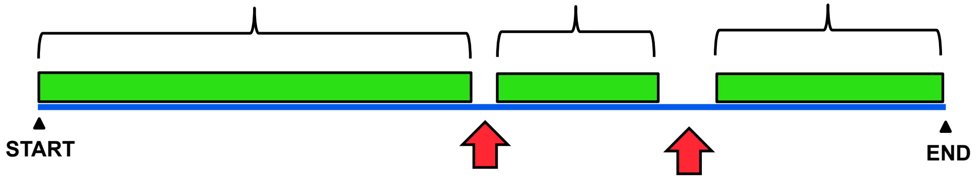
SPK containing data for several objects, each having the same coverage and with no data gaps



SPK File Coverage - 2

- For any request time within any time interval comprising the coverage for an object, the SPK subsystem can return a vector representing the state of that body relative to its center of motion.
  - The SPK system will automatically interpolate ephemeris data to produce a state vector at the request time.
  - To a user's program, the ephemeris data appear to be continuous over each time interval, even if the data stored inside the SPK file are discrete.
- The SPK subsystem will *not* return a result for a request time falling within a data gap.

“Results” will be returned by the SPK reader API for any request time falling within these three intervals



No results will be returned by the SPK reader API for any request time falling within these two data gaps

Reference Frames Used in Writing and Reading SPKs

- All ephemeris data have an associated reference frame
  - The frame specification is provided by the SPK producer

- A program reading an SPK file specifies relative to what reference frame the output state or position vectors are to be given; you're not stuck with using the frame the SPK producer used
  - This output frame must be known to the program
    - » "Known" means either a built-in frame (hard coded in SPICE) or one fully specified at run-time
    - » The user's program may need to have access to additional SPICE data in order to construct the specified frame

Barycenters

- **For planets**
  - A planet and its satellites orbit the planet system's barycenter
    - » For example, the planet Jupiter (599) and each of Jupiter's satellites (501 - 5xx) orbit the Jupiter system barycenter (5)
  - Because Mercury and Venus have no satellites, their barycenters (1 and 2) are at exactly the same locations as their mass centers (199 and 299)
    - » Therefore SPICE ephemeris objects 199 and 299 as well as 1 and 2 are found in a planet ephemeris file
  - Because the masses of Phobos and Deimos are so small compared to the mass of Mars, the mass center for Mars (499) was treated as being located at the Mars barycenter (4)
    - » Starting in 2013 with the JPL planetary ephemeris named DE430 this is no longer the case; there is a very small offset, in the range of 10 to 20 cm

- **For the solar system**
  - Planet system barycenters (i.e. 1 through 9) and the sun (10) orbit the solar system barycenter (0)

Barycenter Offset Magnitude

Navigation and Ancillary Information Facility

Body Mass Center ID	System Barycenter ID	Barycenter offset from body mass center (km)*	Offset as % of body radius*
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)	0.002	~ 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	172%

* Estimated maximum values over the time range 2000-2050

SPK File Ephemeris Contents

- A single SPK file can hold data for one ephemeris object, or for many ephemeris objects

- The objects in a given SPK file need not all be of the same type
  - One might find data for a spacecraft, some planets, and some satellites all in one file, split across multiple segments

- This is illustrated in the next three charts

Examples of Generic SPK File Contents

Planet Ephemeris

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

Asteroid Ephemeris

10	*Sun*
2000001	Ceres

Merged Planet[4] and Satellite Ephemeris

5	*Jupiter BC*
3[4]	Earth BC
10[4]	Sun
399[4]	Earth
501	Io
502	Europa
503	Ganymede
504	Callisto
505	Amalthea
514	Thebe
515	Adrastea
516	Metis
599	Jupiter

Notes:

(1) Mercury and Venus planet locations are included in planet ephemerides since there are no satellite ephemerides for these planets.

(2) The Moon and Earth locations are included in each planetary ephemeris because of historical ephemeris production techniques.

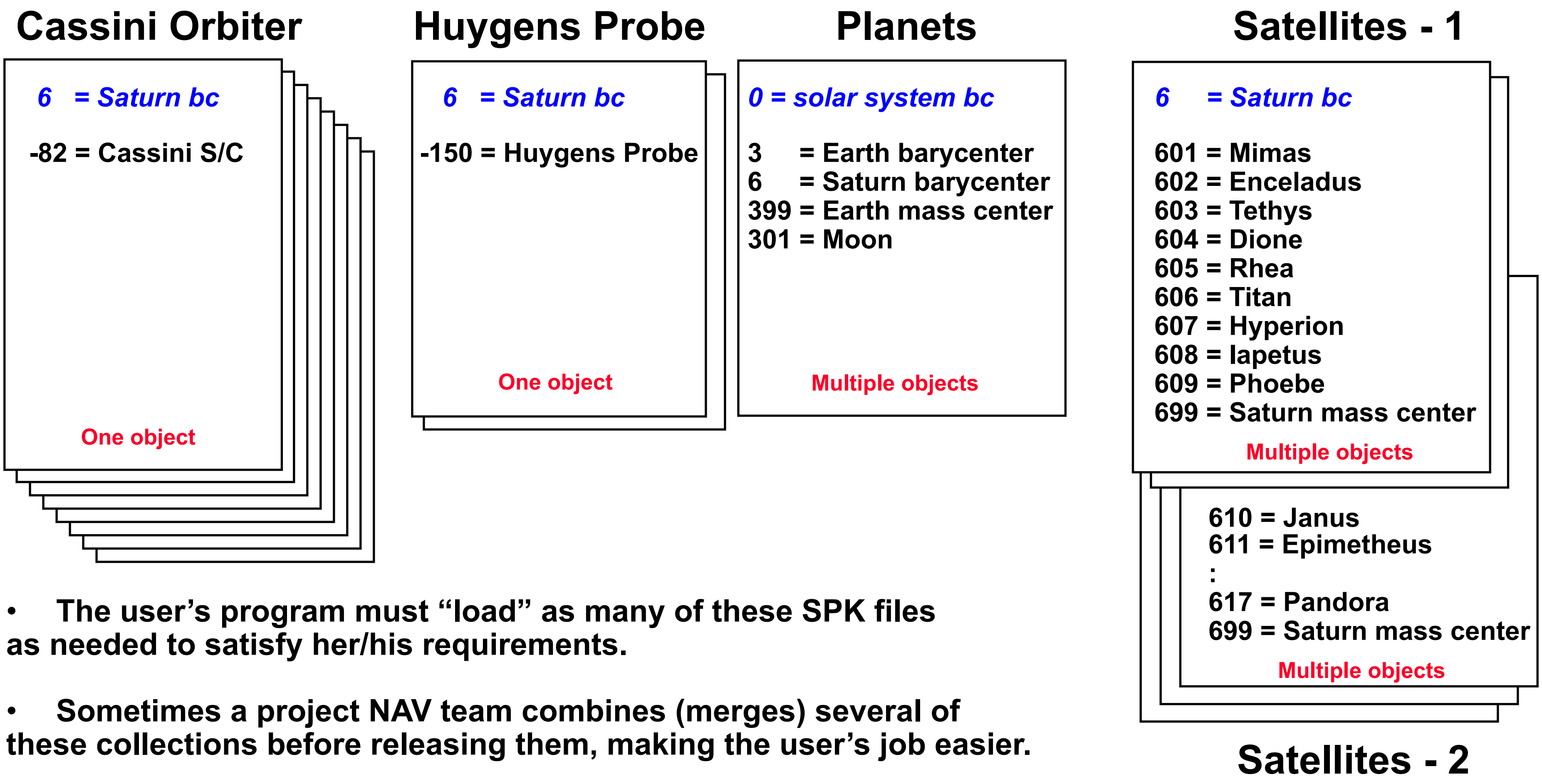
(3) The Sun's location is included in each planetary ephemeris because of historical ephemeris production techniques.

(4) For user convenience, NAIF usually merges into a planet's satellite ephemeris files the locations of the earth, the earth barycenter and the sun.

The objects in blue font are the center of motion for the remaining objects in each file. These is no trajectory data present for these centers of motion.

Examples of Flight Project SPK Files

This made-up example shows four collections of SPK files for the Cassini mission



Cassini Orbiter

6 = Saturn bc

-82 = Cassini S/C

One object

Huygens Probe

6 = Saturn bc

-150 = Huygens Probe

One object

Planets

0 = solar system bc

3 = Earth barycenter
6 = Saturn barycenter
399 = Earth mass center
301 = Moon

Multiple objects

Satellites - 1

6 = Saturn bc

601 = Mimas
602 = Enceladus
603 = Tethys
604 = Dione
605 = Rhea
606 = Titan
607 = Hyperion
608 = Iapetus
609 = Phoebe
699 = Saturn mass center

Multiple objects

Satellites - 2

610 = Janus
611 = Epimetheus
:
617 = Pandora
699 = Saturn mass center

Multiple objects

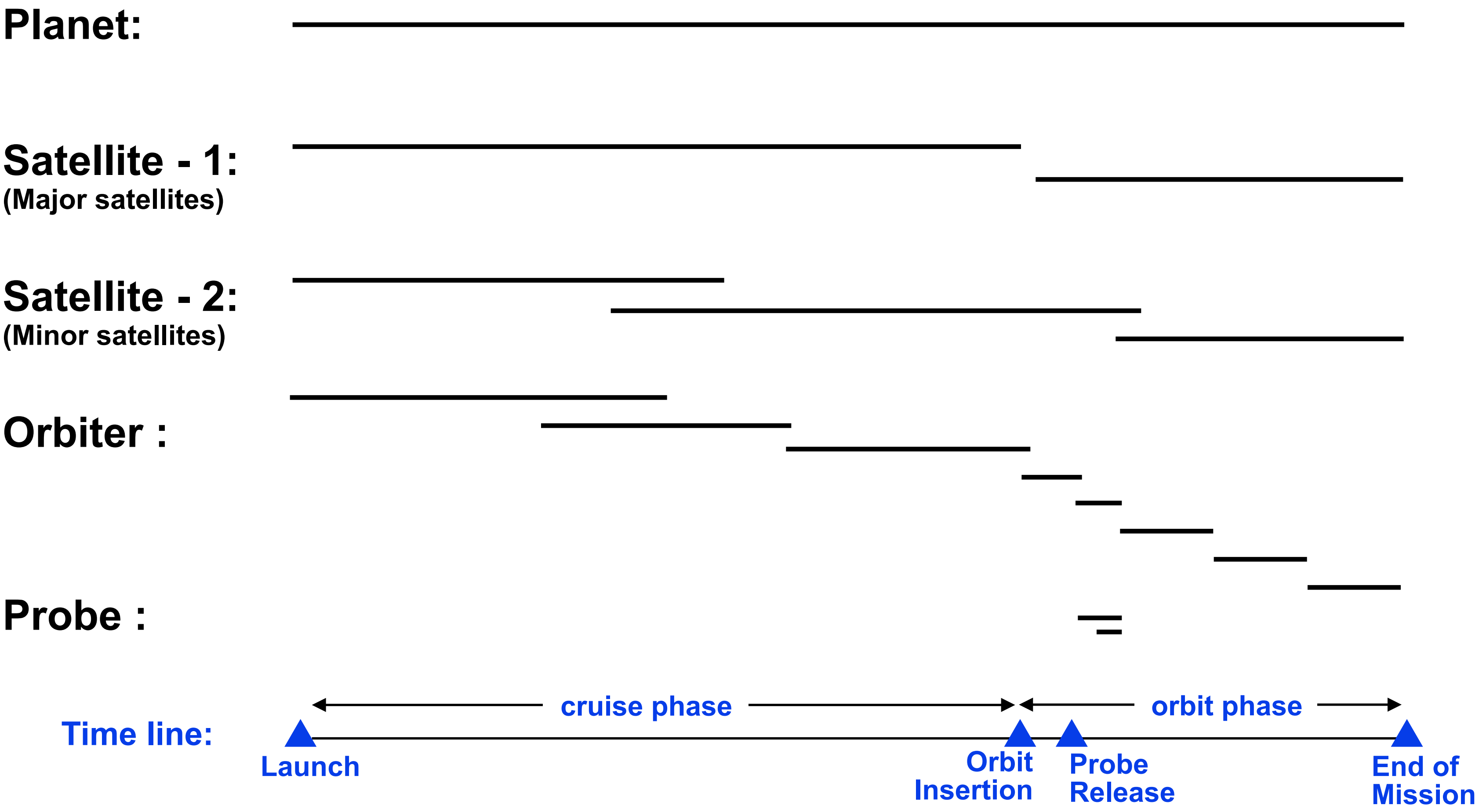
- The user’s program must “load” as many of these SPK files as needed to satisfy her/his requirements.
- Sometimes a project NAV team combines (merges) several of these collections before releasing them, making the user’s job easier.
- *Objects in blue font are the centers of motion for the remaining objects.*

See the next page for a graphical representation of this collection of SPKs

Possible* SPK File Time Coverages for the Previous Example

Navigation and Ancillary Information Facility

Each bar represents a separate file



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

SPKs for Objects Located on the Surface of a Natural Body

- An SPK file may contain positions of tracking stations, observatories, rovers, etc.
    - The object could be stationary or moving
    - Usually such SPKs contain ephemeris data given in the body-fixed reference frame

- 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
        - » Needed to rotate body-fixed vectors to the J2000 frame

Using SPK Files

Retrieving Position or State Vectors

- To retrieve position or state vectors of ephemeris objects one normally needs two kinds of SPICE kernels
  - Ephemeris kernel(s) (SPK)
  - Leapseconds kernel (LSK)
    - » Used to convert between Coordinated Universal Time (UTC) and Barycentric Dynamical Time (TDB, also called Ephemeris Time, ET)
- Retrieving ephemeris data from an SPK file is usually called "reading" the file
  - This term is not very accurate since the SPK "reader" software also performs interpolation, and may chain together data from multiple sources, do frame transformations and perform aberration corrections
- State and position vectors retrieved from an SPK file by the SPK "reader" routines are of the form:
  - X,Y, Z, dX, dY, dZ for a state vector (km, km/sec)
  - X, Y, Z for a position vector (km)

Retrieving a State Vector

Fortran syntax used here

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')

It’s better to 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

Now use the returned state vector in other SPICE routines to compute observation 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.
  - 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 the output state vector is to be given. SPK software will automatically convert ephemeris data to the frame you specified (if needed). SPICE must know the named frame. If it is not a built-in frame SPICE must have sufficient data at run-time to construct it.

* 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 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. ('CN+S' is better in some cases.)
  - 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. The position component of the state vector 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.

LT + S = light time plus stellar aberration
CN + S = converged Newtonian light time plus stellar aberration

A Simple Example of Retrieving a State Vector

Initialization - typically do this just once per program execution

```
CALL FURNSH ( 'NAIF0010.TLS' )
CALL FURNSH ( 'HUYGENS_3_MERGE.BSP' )
```

It’s better to replace these two calls with a single call to a “furnsh kernel” containing the names of all kernel files to load.

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 (which in SPICE is the same as the ICRF 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 (NAIF0010.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.

Retrieving a Position Vector

- SPKPOS is the position-only analog of SPKEZR
  - The arguments of SPKPOS are identical to those of SPKEZR, except that SPKPOS returns a 3-component position vector instead of a 6-component state vector
  - 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 absence of C-kernel angular velocity data

A Slightly More Complex Example - 1
Kernel Data Needed

- To get state vectors 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 body in the body’s IAU body-fixed reference frame you’ll need:
    - PCK file containing orientation data for the body
    - SPK(s) for the object and body
    - 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 a State Vector

Obtain the state of Titan relative to Huygens Probe in the Titan body-fixed reference frame

Initialization...typically once per program execution

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

```
CALL FURNSH  ('HUYGENS_MERGED_SPK.BSP')
CALL FURNSH  ('NAIF0010.TLS')
CALL FURNSH  (’NAIF0010.TPC')
```

It's better to replace these three 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 ( '2004 NOV 21 02:40:21.3', TDB)
```

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

```
CALL SPKEZR (’TITAN', TDB, 'IAU_TITAN’, '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.
```

Manipulating and Using SPK Files

Navigation and Ancillary Information Facility

- You can subset an SPK, or merge two or more SPKs
  - The subset or merge may be keyed 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
- You can convert an SPK that is in non-native binary format to native binary format if you need to add data or comments

* The allowed number of simultaneously loaded DAF-based files is set to 5000 in N65 Toolkits. “DAF” is Double Precision Array File.

Understanding an SPK File

- The SPK producer should have provided descriptive meta-data inside an SPK file, in the “comment area”
  - The comments should say when/why/how and for what purpose the file was made
  - Additional useful information could also be provided by the producer
    - Example: when and why any data gaps are present

- These comments may be extracted or viewed using an API (subroutine) or a SPICE utility program.
  - APIs: SPC…
  - Utility program: commnt -r

SPK Utility Programs

- The following SPK utility programs are included in the Toolkit:

BRIEF	summarizes coverage for one or more SPK files
SPACIT	generates segment-by-segment summary of an SPK file
COMMNT	reads, appends, or deletes comments in an SPK file
MKSPK	converts ephemeris data provided in a text file into an SPK file
SPKDIFF	compares two SPK files
SPKMERGE	subsets or merges one or more SPK files

- These additional SPK utility programs are provided on the NAIF Web site (http://naif.jpl.nasa.gov/naif/utilities.html)

SPY	validates, inspects, and analyses SPK files
PINPOINT	creates an SPK file for fixed locations (ground stations, etc)
BSPIDMOD	alters body IDs in an SPK file
DAFMOD	alters body or frame IDs in an SPK file
DAFCAT	concatenates together SPK files
BFF	displays binary file format of an SPK file
BINGO	converts SPK files between big- and little-endian formats

Summarizing an SPK File

- A summary of the contents and time coverage of an SPK file can be made using the SPICE Toolkit utility *"brief"*
  - See the brief User's Guide for details

```
% brief 070413BP_SCPSE_07097_07121.bsp



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, the default is ET, not UTC!

Additional Information on SPK

- For more information about SPK, look at the following:
  - The on-line (full) SPK tutorial
  - Most Useful Routines document
  - SPK Required Reading document
  - Headers of the subroutines mentioned
  - Using Frames tutorial
  - BRIEF and SPKDIFF User’s Guides
- Related documents:
  - NAIF_IDS Required Reading
  - Frames Required Reading
  - Time Required Reading
  - Kernel Required Reading