



---

Navigation and Ancillary Information Facility

# **“Camera-matrix” Kernel CK**

**(Orientation or Attitude Kernel)**

**Emphasis on reading CK files**

**September 2009**



# CK File Contents - 1

Navigation and Ancillary Information Facility

- **A CK file holds orientation data for a spacecraft or a moving structure on the spacecraft**
  - “Orientation data”  $\Rightarrow$  quaternions, from which orientation matrices are formed by SPICE software. These matrices are used to rotate position vectors from a base reference frame (the “from” frame) into a second reference frame (the “to” frame)
    - » In SPICE this is often called the “C-matrix, short for “Camera matrix”
  - Optionally may include angular velocity of rotation of the “to” frame with respect to the “from” frame
    - » Angular velocity vectors are expressed relative to the “from” frame.
- **A CK file should also contain comments—sometimes called metadata—that provide some details about the CK such as:**
  - the purpose for this particular CK
  - when and how it was made
  - what time span(s) the data cover



## CK File Contents - 2

---

Navigation and Ancillary Information Facility

- **A single CK file can hold orientation data for one, or for any combination of spacecraft or their structures**
  - **Some examples**
    1. Huygens Probe
    2. Cassini Orbiter and its CDA instrument mirror
    3. Mars Express Orbiter, PFS scanner, Beagle Lander
    4. MRO orbiter, MRO high gain antenna, MRO solar arrays
- **In practice CKs usually contain data for just one structure**

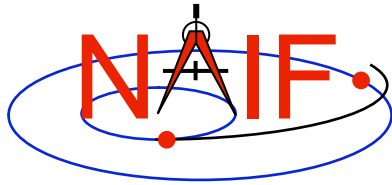


# CK File Varieties

---

Navigation and Ancillary Information Facility

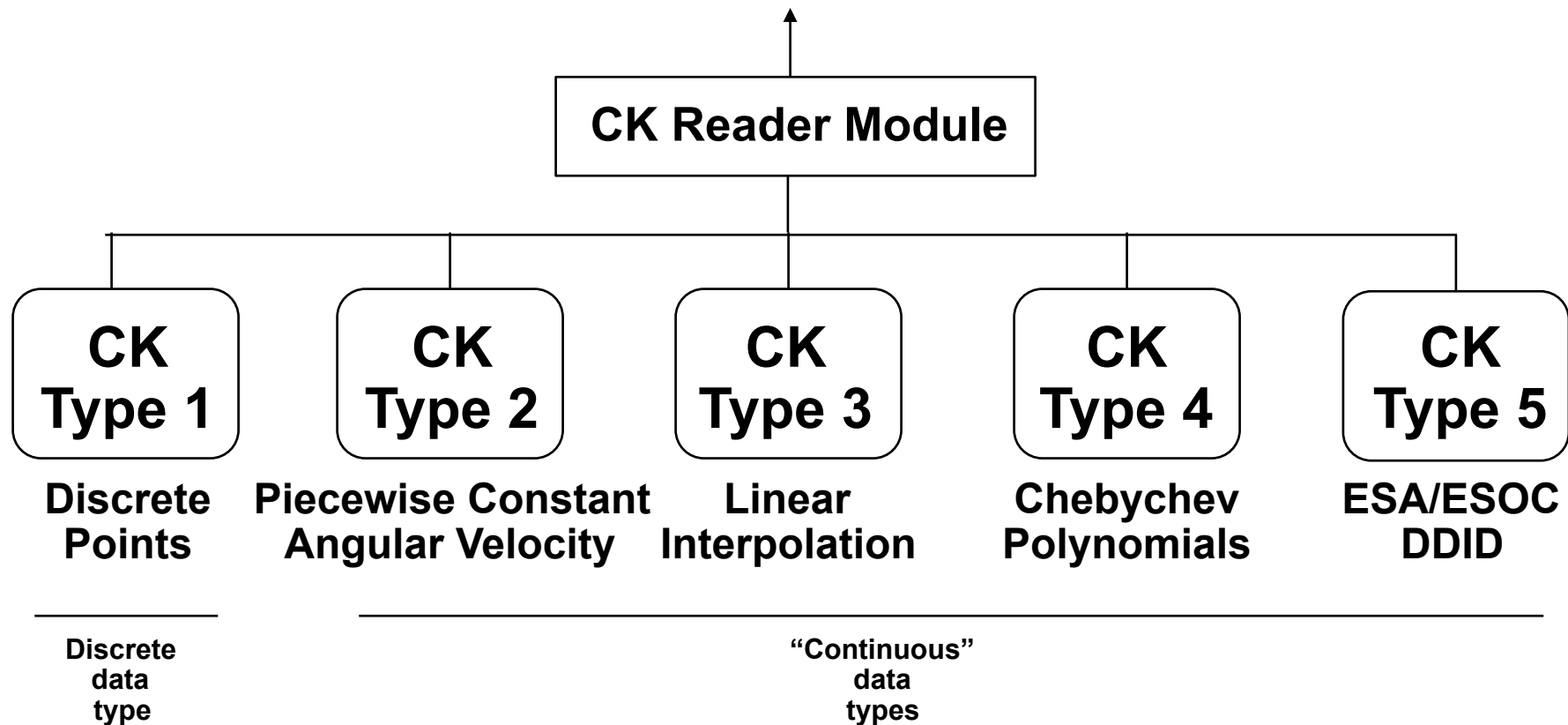
- **“Reconstruction” CK (also called “definitive” CK)**
  - A CK file can be made from orientation telemetry returned from a spacecraft or other structure
  - A CK might also be made from some process that improves upon the pointing determined from downlinked telemetry (“C-smithing”)
  - These kinds of files are generally used for science data analysis or spacecraft performance analysis
- **“Predict” CK**
  - A CK file can be made using information that predicts what the orientation will be some time in the future
    - » Input data usually come from a modeling program, or a set of orientation rules
  - This kind of file is generally used for engineering assessment, science observation planning, software testing and quick-look science data analysis
    - » If it has good fidelity, such a file might be used to “fill in the data gaps” of a reconstruction CK file
    - » In some cases (ESA in particular) the predict meets the reconstruction accuracy requirements; a true reconstruction CK is not produced

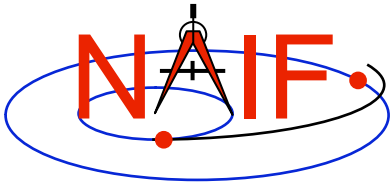


# CK Data Types Concept

Navigation and Ancillary Information Facility

The underlying orientation data are of varying types, but the user interface to each of these CK types is the same.





# Kernel Data needed

---

Navigation and Ancillary Information Facility

- To get orientation (rotation matrix) and angular velocity of a spacecraft or one of its moving structures, one needs at least three SPICE kernel types: CK, SCLK and LSK. One may also need an FK, if he or she plans to access CK data via high level SPICE interfaces.
  - CK contains spacecraft or other structure orientation
  - SCLK and LSK contain time correlation coefficients used to convert between ephemeris time (ET) and spacecraft clock time (SCLK)
    - » Sometimes an LSK is not needed in this conversion, but best to have it available as it is usually needed for other purposes
  - FK associates reference frames with CK IDs
    - » Needed if high level SPICE interfaces are used to access CK data (see next page)



# What SPICE Routines Access CKs?

Navigation and Ancillary Information Facility

- High-level SPICELIB routines are used more often than the “original” CK readers to access CK data. These high-level routines are:
  - Position or state transformation matrix determination
    - » **PXFORM**: return a rotation matrix (3x3) from one frame to another, either of which can be a CK-based frame or have CK-based frames as “links” in its chain
    - » **SXFORM**: return a state transformation matrix (6x6) from one frame to another, either of which can be a CK-based frame or have CK-based frames as “links” in its chain
  - Position or state vector determination
    - » **SPKPOS**: return a position vector (3x1) in a specified frame, which can be a CK-based frame or have CK-based frames as “links” in its chain
    - » **SPKEZR**: return a state vector (6x1) in a specified frame, which can be a CK-based frame or have CK-based frames as “links” in its chain
- Use of the above mentioned routines is discussed in the FK, Using Frames, and SPK tutorials
- The “original” CK access routines are CKGP and CKGPAV
  - Use of these routines is described in this tutorial, along with topics applicable to all of the routines mentioned above



## Initialization ... typically once per program run

```
CALL FURNISH( 'lsk_file_name' )
CALL FURNISH( 'sclk_file_name' )
CALL FURNISH( 'ck file name' )
```

## Loop ... do as often as you need

```
CALL STR2ET( 'utc_string', tdb )
CALL SCE2C ( spacecraft id, tdb, sclkdp )
```

```

or CALL CKGP      (instid,sclkdp,tol,'ref_frame',cmat,  clkout,found)
CALL CKGPAV      (instid,sclkdp,tol,'ref frame',cmat,av,clkout,found)

```

## outputs

8





# Arguments of CKGPAV

Navigation and Ancillary Information Facility

## Inputs

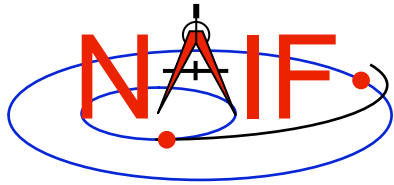
<b>instid</b>	NAIF ID for the spacecraft, instrument or other structure for which the orientation is to be returned
<b>sclmdp</b>	the time at which the orientation matrix and angular velocity are to be computed. The time system used is encoded spacecraft clock time (SCLK). The units are ticks since the zero epoch of the clock
<b>tol*</b>	the tolerance, expressed as number of SCLK ticks, to be used in searching for and computing the orientation data
<b>ref_frame</b>	the name of the reference frame with respect to which the orientation is to be computed. This is also called the “base” or “from” frame.

## Outputs

<b>cmat</b>	the 3x3 rotation matrix that you requested
<b>av</b>	the angular velocity that you requested
<b>clkout</b>	the exact time for which the orientation and angular velocity was computed
<b>found</b>	the logical flag indicating whether the orientation and angular velocity data were found. Note that if the loaded CK file(s) do not contain angular velocity data, CKGPAV will return a FALSE found flag even if orientation could have been computed. If “found” is .FALSE., then the values of the output arguments “cmat” and “av” are undefined and <b>should not be used!</b>

**Always check the FOUND flag returned by CKGPAV and CKGP!**

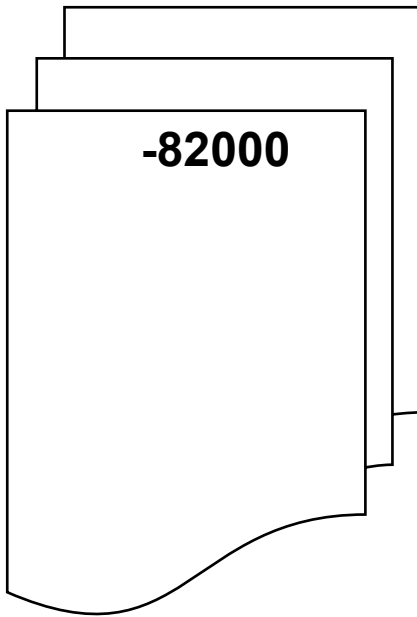
\* tol is explained in detail later on



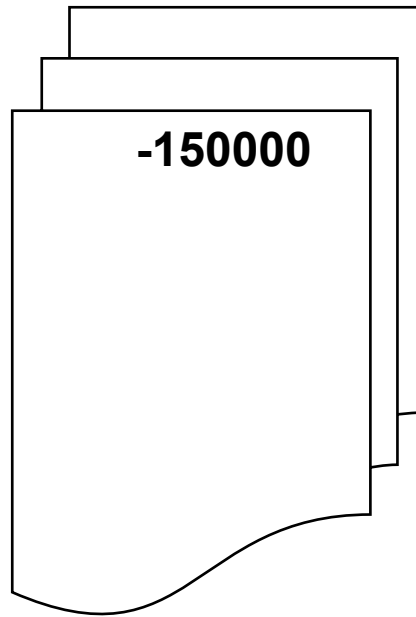
# An Example of Project CK Files

Navigation and Ancillary Information Facility

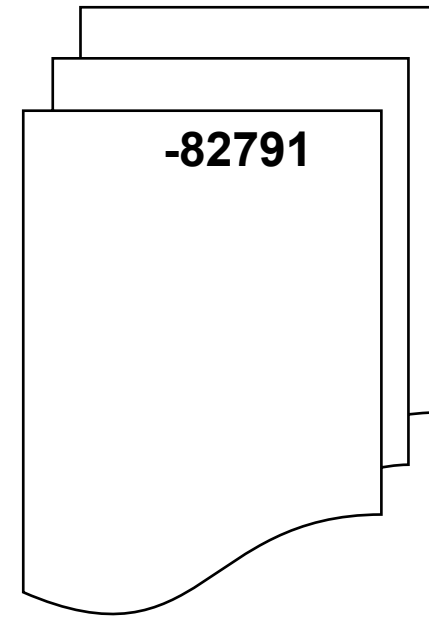
## CASSINI SPACECRAFT



## HUYGENS PROBE



## CASSINI CDA MIRROR



**A user's program must be able to load as many of these files as needed to satisfy his/her requirements. It is strongly recommended that such programs have the flexibility to load a list of CK files provided to the program at run time; this is easily accomplished using the Toolkit's FURNISH routine.**



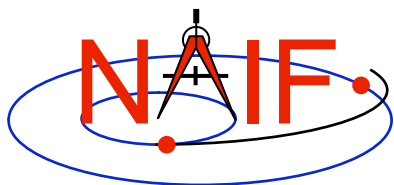
A black line graph on a white background. The line starts at the top left and descends in a series of steps, ending at the bottom right. The horizontal segments of the line are of varying lengths, and the vertical drops are also of varying lengths, creating a jagged, descending staircase effect.

**[REDACTED]**

Age Group	Percentage of Respondents
18-29	85
30-39	75
40-49	65
50-59	55
60-69	45
70-79	35
80+	15

▲ Launch ← cruise phase → ▲ Orbit Insertion ▲ Probe Release ▲ End of Mission

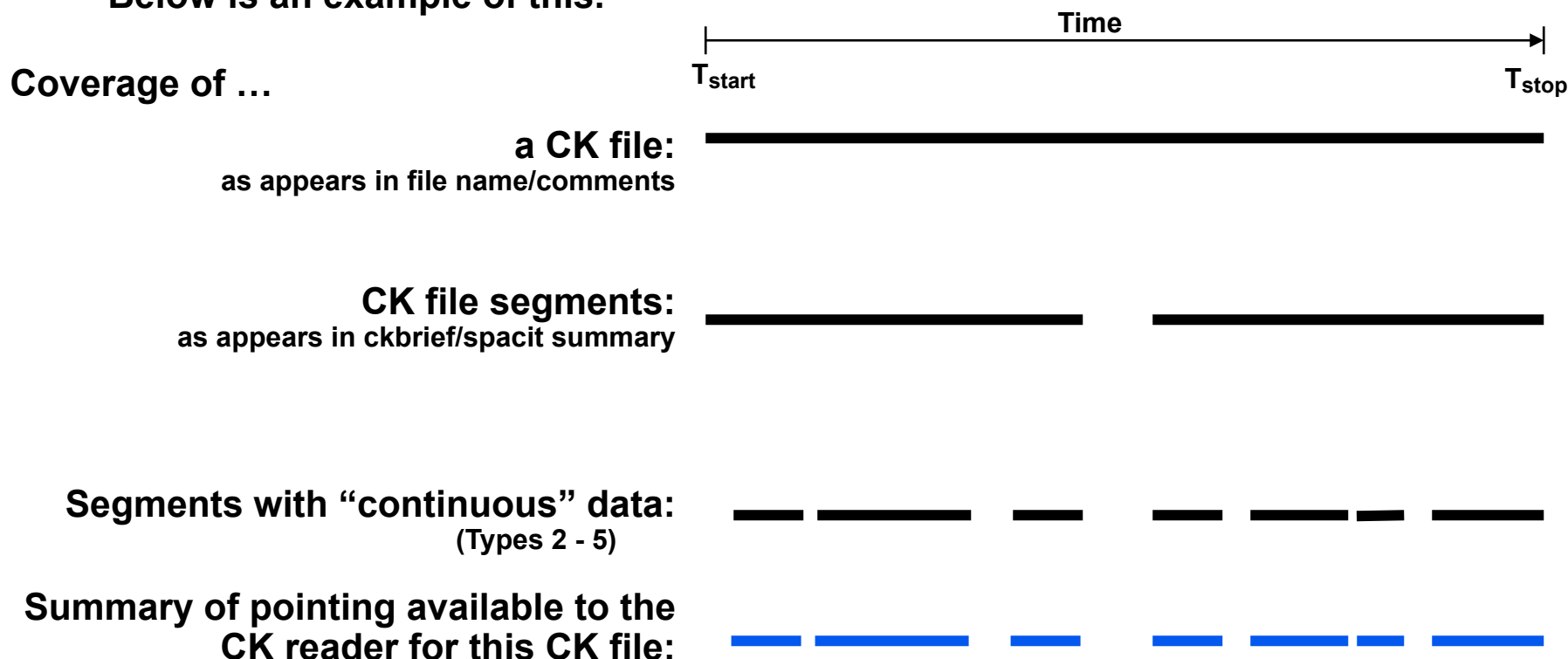
11



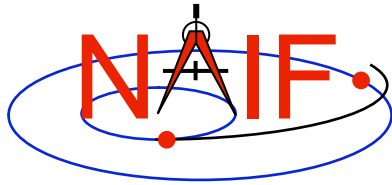
# Sample CK Data Coverage - 2

Navigation and Ancillary Information Facility

Even though a project's CK production process may suggest that CK files provide continuous coverage for the interval of time for which they were generated, in reality this is rarely the case. CK files almost always contain gaps in coverage. Below is an example of this.



Blue line segments represent interpolation intervals—times when pointing will be returned and the FOUND flag set to “TRUE.”



# What is an Interpolation interval?

---

Navigation and Ancillary Information Facility

- **An interpolation interval is a time period for which the CK reader routines can compute and return pointing.**
  - For CK Types 3 and 5 the pointing is computed by interpolating between the attitude data points that fall within the interval.
  - For CK Type 2 the pointing within each interval is computed by extrapolating from a single attitude and associated angular velocity.
  - For CK Type 4 the pointing is computed by evaluating polynomials covering the interval.
  - For CK Type 1 (discrete pointing instances) the notion of an interpolation interval is not relevant.
- **The time periods between interpolation intervals are gaps during which the CK readers are not able to compute pointing.**
- **The interpolation intervals in Type 3 CK segments can be modified without changing the actual pointing data.**
  - The CKSPANIT and CKSMRG programs are used to make these changes.



# The Meaning of Tolerance - 1

---

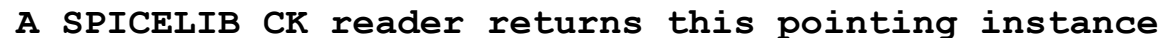
Navigation and Ancillary Information Facility

- The CKGP and CKGPAV routines use a time tolerance, “tol,” measured in ticks, in executing pointing lookups.
  - No matter whether your CK is a discrete type (Type 1) or a continuous type (Types 2 - 5), if pointing information is not found within +/- tol of your pointing request time, no pointing will be returned and the “found flag” will be set to “FALSE.”
  - For Type 1 (discrete) CKs, the pointing instance **nearest\*** to your request time will be returned, as long as it is within tol of your request time.
    - » If the nearest pointing instances on each side of your request time are equidistant from your request time, the later instance will be selected.
  - For Types 2 - 5 (continuous pointing), pointing for **exactly** your request time will be returned if this time falls anywhere within an interpolation interval.
  - For all Types, the time tag associated with the pointing data will also be returned.

\*Ignoring segment priority



## When reading a Type 1 CK containing **discrete** pointing instances



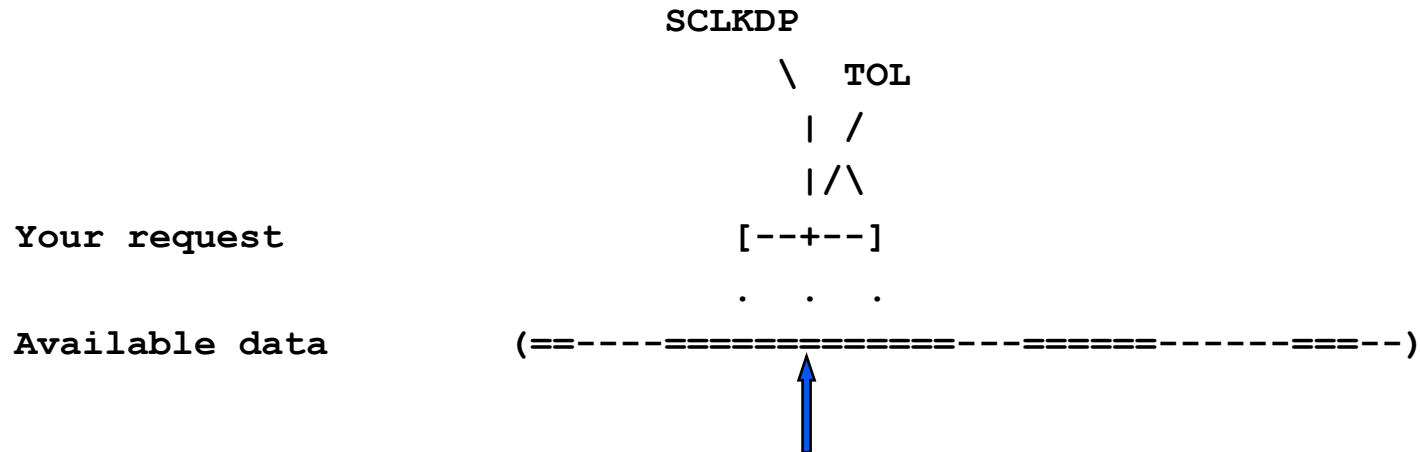
- “0” is used to represent discrete pointing instances (quaternions)
- “ ( ) ” are used to represent the end points of a segment within a CK file
- SCLKDP is the time at which you have requested pointing
- TOL is the time tolerance you have specified in your pointing request
- The quaternions occurring in the segment need not be evenly spaced in time



# The Meaning of Tolerance - 3

Navigation and Ancillary Information Facility

When reading a Type 2, 3, 4 or 5 CK (**continuous pointing**), with a “pointing request” that falls within a span of continuous pointing (an “interpolation interval”)



A SPICELIB CK reader returns pointing at precisely the requested epoch

- “==” is used to indicate interpolation intervals of continuous pointing
- “ ( ) ” are used to represent the end points of a segment within a CK file
- SCLKDP is the time at which you have requested pointing
- TOL is the time tolerance you have specified in your pointing request; for this particular case it does not come into play
- The quaternions occurring in the periods of continuous pointing need not be evenly spaced in time

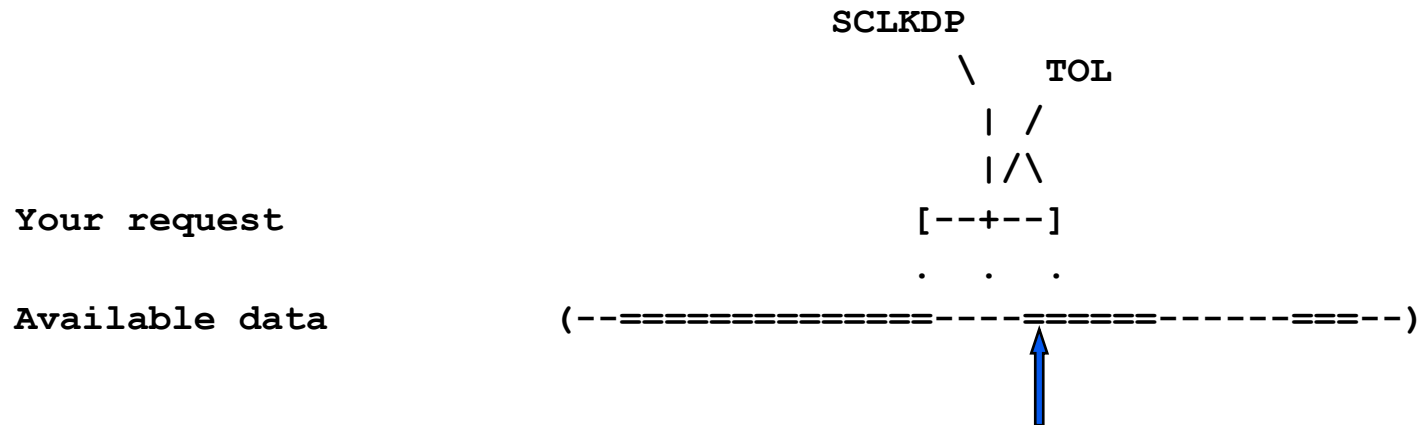




# The Meaning of Tolerance - 4

Navigation and Ancillary Information Facility

When reading a Type 2, 3, 4 or 5 CK (continuous pointing), with a “pointing request” that is NOT within a span of continuous pointing (an “interpolation interval”)



A SPICELIB CK reader returns pointing at the epoch closest to the request time, if this is within TOL of that request time.

- “==” is used to indicate interpolation intervals of continuous pointing
- “ ( ) ” are used to represent the end points of a segment within a CK file
- SCLKDTP is the time at which you have requested pointing
- TOL is the time tolerance you have specified in your pointing request
- The quaternions occurring in the periods of continuous pointing need not be evenly spaced in time



# Summarizing a CK file - 1

---

Navigation and Ancillary Information Facility

- A brief summary of a CK can be made using the Toolkit utility **CKBRIEF**

- At your command prompt, type the program name followed by the names of CK, LSK and SCLK files (given in any order)

```
% ckbrief 07102_07107ra.bc naif0008.tls cas00106.tsc
```

```
CKBRIEF Version: 3.2.0, 2006-11-02. SPICE Toolkit  
Version: N0061.
```

```
Summary for: 07102_07107ra.bc
```

```
Object: -82000
```

Interval Begin ET	Interval End ET	AV
2007-APR-12 00:01:06.462	2007-APR-17 00:01:03.726	Y



## Summarizing a CK file - 2

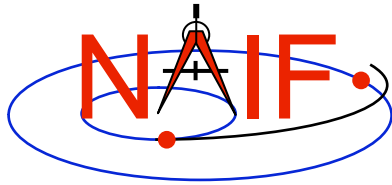
---

Navigation and Ancillary Information Facility

- A detailed summary of a CK can be made using the Toolkit utility SPACIT. See the SPACIT User's Guide for details.

```
-----  
Segment ID      : CASSINI ATT: -Y TO TITAN, +Z - VELCTY  
Instrument Code: -82000  
Spacecraft      : Body -82, CAS  
Reference Frame: Frame 1, J2000  
CK Data Type    : Type 3  
  Description   : Continuous Pointing: Linear Interpolation  
Available Data  : Pointing and Angular Velocity  
UTC Start Time  : 2005 FEB 15 07:59:59.999  
UTC Stop Time   : 2005 FEB 15 08:59:59.998  
SCLK Start Time: 1/1487147147.203  
SCLK Stop Time  : 1/1487150747.209  
-----
```

```
      .      .  
      .      .  
    etc.    etc.
```



# Obtaining Coverage of CK File

---

## Navigation and Ancillary Information Facility

- **High-level subroutine interfaces for obtaining CK coverage information were introduced in the N0058 version of the SPICE Toolkit.**
  - **Call CKOBJ to find the set of structures for which a specified CK provides data.**
    - » **INPUT:** an CK 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 structures for which the specified CK provides coverage.

**CALL CKOBJ ( CK, IDS )**

- **Call CKCOV to find the window of times for which a specified CK file provides coverage for a specified structure:**
  - » **INPUT:** a CK file name, structure ID code, flag indicating whether angular rates are needed, flag indicating whether coverage on segment or interval level is to be returned, tolerance, output time system, 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 coverage intervals of the specified CK, expressed as encoded SCLK or ET seconds past J2000.

**CALL CKCOV ( CK, ID, NEEDAV, LEVEL, TOL, TIMSYS, 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.**



# Make Use of CKCOV

---

Navigation and Ancillary Information Facility

- **When using high-level routines\* that need orientation data from a C-kernel, it's often a good idea to first determine what are the valid interpolation intervals in your CK using CKCOV.**
  - If using multiple CKs, all of which are needed to construct a frame chain, call CKCOV for each one and then intersect the coverage windows.
- **Then check each time of interest for your geometry calculations against the window of valid intervals before proceeding onwards.**

\* E.g. SPKEZR, SPKPOS, SXFORM, PXFORM, SINCPT



# CK Utility Programs

---

Navigation and Ancillary Information Facility

- **The following CK utility programs are included in the Toolkit:**

<b>CKBRIEF</b>	<b>summarizes coverage for one or more CK files</b>
<b>SPACIT</b>	<b>generates segment-by-segment summary of a CK file</b>
<b>COMMNT</b>	<b>reads, appends, or deletes comments in an CK file</b>
<b>MSOPCK</b>	<b>converts attitude data provided in a text file into a CK file</b>
<b>FRMDIFF</b>	<b>samples or compares orientation of CK-based frames</b>

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

<b>CKSLICER</b>	<b>subsets a CK file</b>
<b>CKSMRG</b>	<b>merges segments in a type 3 CK file (*)</b>
<b>DAFCAT</b>	<b>concatenates together CK files (*)</b>
<b>CKSPANIT</b>	<b>modifies interpolation interval information in a Type 3 CK file</b>
<b>DAFMOD</b>	<b>alters structure or frame IDs in a CK file</b>
<b>prediCkt</b>	<b>creates a CK file representing an orientation profile described by a set of orientation rules and a schedule</b>
<b>BFF</b>	<b>displays binary file format of an CK file</b>
<b>BINGO</b>	<b>converts CK files between IEEE and PC binary formats</b>

(\*) DAFCAT and SKSMRG are frequently used together to first merge many CK files into a single file using DAFCAT and then merge segments inside the merged file using CKSMRG.

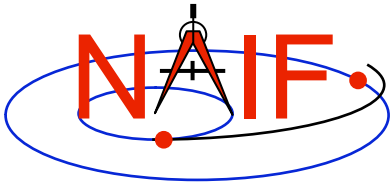


# Additional Information on CK

---

Navigation and Ancillary Information Facility

- **For more information about CK, look at the following documents**
  - CK Required Reading
  - headers for the CKGP and CKGPAV routines
  - Most Useful SPICELIB Routines
  - CKBRIEF and FRMDIFF User's Guides
  - Frames tutorials: FK and Using Frames ← *don't miss these*
  - Porting\_kernels tutorial
- **Related documents**
  - SCLK Required Reading
  - Time Required Reading
  - Frames Required Reading
  - NAIF\_IDS Required Reading
  - Rotations Required Reading



---

Navigation and Ancillary Information Facility

## Backup

### Examples of Problems Encountered When Using the CK Subsystem





# Problems using CK - 1

---

Navigation and Ancillary Information Facility

- **The file or files you loaded do not contain orientation data for the object of interest.**
  - Make sure the ID that you use in a call to CKGP or CKGPAV matches one in the CK file(s) you have loaded.
  - Make sure the frame that you specify in a call to SXFORM, PXFORM, SPKEZR, or SPKPOS is transformable to one available in the loaded CK files.
- **CKGP or CKGPAV returns a transformation matrix and/or angular velocity that does not appear correct.**
  - Probably the FOUND flag is “FALSE” and you are using data left over from a previous query. Remember to **always check the FOUND flag!** (If the FOUND flag is “TRUE” but the data seem bad, contact the data producer.)



## Problems using CK - 2

---

Navigation and Ancillary Information Facility

- **The file, or files, you loaded do not cover the time at which you requested orientation**
  - Check file coverage on the segment level by summarizing the file (s) using CKBRIEF or SPACIT
  - Check interpolation interval coverage using CKBRIEF with option “-dump,” or by examining comments provided in the comment area of the file - you may be asking for data within a coverage gap (outside of interpolation intervals)
- **One of the frames routines (SPKEZR, SPKPOS, SXFORM, PXFORM, SINCPT) signals an error**
  - All frames routines read CK files using tolerance = 0
    - » For discrete CKs (Type 1) the orientation of a CK-based frame will be computed only if the time provided to a Frames routine exactly matches one of the times stored in the CK file; otherwise an error will be signaled.
    - » For continuous CKs (all but Type 1) the orientation of a CK-based frame will be computed only if the time provided to a frame routine falls within one of the interpolation intervals defined by the CK file; otherwise an error will be signaled.



## Problems using CK - 3

---

Navigation and Ancillary Information Facility

- **You've confirmed not having any of the previously described problems, but the FOUND flag comes back "FALSE" when using CKGPAV, or SXFORM or SPKEZR signals a frame related error.**
  - **You are using a SPICE routine that expects angular velocity as well as orientation, but the CK segments available at your requested epoch don't contain angular velocity.**
    - » **Routines expecting AV are: CKGPAV, SXFORM, SPKEZR**
    - » **Routines not expecting AV are: CKGP, PXFORM, SPKPOS**



## Problems using CK - 4

---

Navigation and Ancillary Information Facility

- The FOUND flag comes back “TRUE” when using CKGPAV but returned angular velocity does not appear correct.
  - While many other sources of the angular rate data, for example spacecraft telemetry, specify it relative to the spacecraft frame, SPICE CK files store it, and CKGPAV returns it, with respect to the base frame. So the CK style of returned angular velocity may be unexpected.
- The FOUND flag comes back “TRUE” when using CKGP/CKPGAV but the quaternion computed from the returned transformation matrix via a call to M2Q does not appear correct.
  - The quaternion returned by M2Q follows the SPICE style, which is different from the quaternion styles used by some other sources of orientation data, for example most spacecraft telemetry.
    - » See the headers of the M2Q and Q2M routines, and the Rotations Required reading document for more details.
    - » NAIF also prepared and can provide a “white paper” explaining differences between various quaternion styles commonly used in space applications.



## Problems using CK - 5

---

Navigation and Ancillary Information Facility

- **You're trying to use a CK file that is not properly formatted for your computer**
  - You can read only a binary CK file with the CK subroutines; you can't read a "transfer format" file
    - » Although not required, binary CK files often have a name like "xxxxxx.bc"
    - » Although not required, transfer format CK files often have a name like "xxxxxx.xc"
  - If using Toolkit Version N0051 or earlier you must have the proper kind of CK binary file for your computer (a native binary file)
    - » Sun, HP, SGI and MAC (Motorola cpu) use the unix binary standard
    - » PC (Windows or Linux) uses its own binary standard
    - » The pair of utility programs named TOBIN and TOXFR, or the utility program SPACIT, can be used to port CK files between computers having incompatible binary standards