PCK REQUIRED READING

Navigation Ancillary Information Facility

Prepared by
E. D. Wright

NAIF Document No. 254.04
Last Modified: 06 OCT 1999

Table of Contents

PCK

Revisions

Purpose

The purpose of this document is to describe both the format and contents of the "Planetary Constants" kernel files (PCKs) used within the SPICELIB system, and to describe the NAIF Toolkit software that reads and makes direct use of these kernel files.

Intended Audience

This document is recommended reading for all users of PCK files.

References

All references are to NAIF documents. The notation [Dn] refers to NAIF document numbers.

1. [218] KERNEL Required Reading.
2. [219] NAIF IDS Required Reading.
3. [168] SPK Required Reading.
4. [225] TIME Required Reading.
5. [214] ROTATIONS Required Reading.
6. [167] Double Precision Array Files (DAF) Required Reading.
7. [195] "Planetary Geodetic Control Using Satellite Imaging," Journal of Geophysical Research, Vol. 84, No. B3, March 10, 1979, by Thomas C. Duxbury.
8. [306] "Report of the IAU/IAG/COSPAR Working Group on Cartographic Coordinates and Rotational Elements of the Planets and Satellites: 1991," March 3, 1992.

Introduction

The "P" in "SPICE" has two sub-components: PCK kernels, which contain planetary attitude and body shape information, and SPK kernels, which contain planetary ephemerides. The PCK kernel includes orientation and shape information pertaining to extended solar system bodies such as the Sun, the planets, the natural satellites of the planets, comets, and asteroids.

The functionality of the PCK kernel is supplied by data files called "PCK kernel files" and by SPICELIB subroutines that can read and interpret the data in these files. Historically, there was only one kind of PCK kernel file, the PCK text kernel file (called the "P_constants kernel.") This kind of PCK kernel file contains constants which allow calculation of solar system body orientation and shape. These ASCII files can be easily reviewed and modified. Now, in addition to these text PCK kernels, another format, the binary PCK kernel, has been developed. These files allow higher accuracy calculations of body orientation. They contain shorter-term, but more precise information in binary format where large amounts of data can be stored and accessed quickly. The only bodies for which this higher accuracy type of data is available are the moon, the Earth and the asteroid Eros.

The purpose of the PCK kernels and associated software is to provide SPICE users with a convenient mechanism for supplying planetary physical constants to application programs. SPICELIB's PCK software is able to read files conforming to these formats and return to a calling application program both the data contained in such files and a small number of commonly used numeric quantities derived from the kernel data.

This document specifies both the SPICELIB PCK formats and the interfaces of the SPICELIB subroutines that read or return data from PCKs. Examples of PCK file contents and code examples demonstrating use of the SPICELIB PCK subroutines are included.

Prerequisites

This section covers some preliminary information required for understanding the subsequent discussion of the contents of PCK files and SPICELIB's PCK software.

Body Codes

Within the NAIF Toolkit software, a system of integer codes is used to conveniently represent both celestial bodies and spacecraft. This system is described in detail in "NAIF IDS Required Reading" [218].

In this document, the following features of the code system will be relied on:

- The code for the barycenter of the nth planetary system is n. The count starts at 1, which stands for Mercury. (The code for the Sun is 10.)
- The code for the nth planet's mass center is n99; for example, the code for the Earth is 399.
- Natural satellites of the nth planet have codes of the form nxx; for example, the code for the Earth's moon is 301.

Epochs and Reference Frames

Some constants that frequently appear in PCK files are associated with a particular epoch and with a particular reference frame. For example, PCK files released by NAIF typically contain constants that define the axes of various body-fixed planetocentric coordinate systems, given relative to a specified inertial reference frame, as a function of time. In this sort of definition, the independent variable—time—is measured relative to a specified reference epoch.

Typical choices of reference frames associated with PCK constants would be J2000 or B1950. Typical choices of reference epochs would be the J2000 epoch (JED 24515145.0) or the J1950 epoch (JED 2433282.5).

Within SPICELIB, reference frames are usually indicated by short character strings, such as 'J2000'. However, SPICELIB also has a system of integer codes that are used by some routines to name reference frames. This coding system is described in detail in [219]. An example of the correspondence of codes and names is show below:

```
Code    Name       Description
-----   --------   --------------------------------------------
  1     J2000      Earth mean equator, dynamical equinox of J2000
  2     B1950      Earth mean equator, dynamical equinox of B1950
```

In PCK files, these integer codes are used to name reference frames associated with constants.

Planetocentric Coordinates

The body-fixed "Planetocentric" coordinate system referred to in this document is defined for solar system bodies as follows:

- The x-axis of the Planetocentric coordinate system for a specified body lies both in the body's equatorial plane and in the plane containing the body's prime meridian.
- The z-axis is parallel to the body's mean axis of rotation and points North of the invariable plane of the solar system (regardless of the body's spin direction). The north pole is this pole of rotation.
- The y-axis is defined as the cross product of the z and x axes, in that order. Thus, the frame is right-handed.

The above definition implies that the axes of a planetocentric system are time-varying. Thus a complete specification of the axes requires identification of an epoch as well as the body. This frame is relative to J2000.

The Two Formats of PCK files

There are two general forms for PCK files, text and binary files. Text files are ASCII and can be created and modified with an editor. Therefore, they are easily changed and read. Binary files are created via SPICELIB programs and have a particular format and architecture. They cannot be examined or changed with an editor. These files require SPICELIB software for their manipulation. Binary files can contain more data and are faster to use. In the PCK case, the binary files contain higher precision data than the text files.

Text PCK Files

Text PCK files express data as "assignments"; in text PCKs, values are associated with name strings using a "keyword = value" format. These name strings, together with their associated values, are called "kernel variables." The SPICELIB routines that access text PCK data at run time use these associations established by loaded text PCK files to reference desired data values; these routines look up data "by name." Therefore, programmers writing applications that use text PCKs must coordinate use of kernel variable names between their software and the text PCK files used by their software.

Binary PCK Files

The binary PCK file format is built upon the SPICELIB system's DAF (Double precision Array File) architecture. Readers who are not familiar with this architecture are referred to [167] which describes the common aspects of all DAF formats, as well as a collection of SPICELIB subroutines that support the DAF architecture. Those users who only intend to read binary PCK files need not read [167].

Using the PCK System: Overview

This section describes how PCK files and software are used in application programs.

Using PCK data in an application program requires three steps:

1. Selecting the appropriate PCK file(s) for the application.
2. Reading the PCK data into the program.
3. Accessing the data within the program.

Step 1 is not necessarily trivial since there may be no single set of "best values" for physical constants of interest; the values that are "best" - if such exist - depend on the application. The user's judgement, supported by comments and usage notes in the PCK file, is required for this step.

Step 2 is referred to as "loading" a PCK file. Text PCK files are loaded by calling the SPICELIB subroutine LDPOOL and supplying the name of the PCK file to load as the input argument. All of the data in a text PCK file is read into memory when the file is loaded by an application program at run-time. Binary PCK files are loaded by calling the SPICELIB subroutine PCKLOF. This data will be accessible throughout the rest of the program run, unless it is deliberately overwritten or unloaded. Multiple text and multiple binary PCK kernels can be used simultaneously.

Binary PCK files take precedence over text PCK files. If data for a requested planetary constant and time period is covered by a loaded binary PCK file, the binary data will be returned and used. If multiple binary PCK files are loaded, the most recently loaded file takes precedence, down to the binary file loaded earliest. Only if no binary PCK data is available will the text PCK file be used.

Step 3, accessing loaded PCK data, is accomplished via calls to SPICELIB routines. At the lowest level, these access routines allow the calling program to retrieve specified data that has been read from one or more PCK files. Higher-level access routines can return quantities derived from PCK data that has been loaded; the most commonly used such quantity is a matrix that transforms vectors from inertial to body-fixed planetocentric coordinates. The SPICELIB subroutine TIPBOD calculates this transformation. The subroutine BODVAR retrieves information on body orientation and body shape by calling RTPOOL to fetch it from the text PCK kernels loaded into the kernel pool.

For text PCK files, the PCK software can be thought of as "buffering" all of the data in the PCK files that have been loaded: all of the data from these files is retained in memory. Therefore, repeated calls to the PCK access routines do not incur the inefficiency of re-reading data from files. For binary PCK file, like the case of the SPK and CK readers, only a portion of the most recently used information is buffered.

The data structure used by SPICELIB to maintain associations of text kernel variable names and values is called the "kernel pool." Data loaded into memory via LDPOOL is referred to as "being present in the kernel pool." There is no analog to the kernel pool for binary PCK files.

Porting PCK Files Between Computers

A NAIF Toolkit utility program (SPACIT) is used to convert binary PCK files to NAIF transfer files for porting to another type of computer; on the new host the SPACIT utility is used to convert the NAIF transfer file to the native binary format. Text format PCK files may be moved between computers using normal file transfer methods such as ftp (text mode), COPY or e-mail.

Orientation Models used by PCK Software

The orientation models used by SPICELIB PCK access routines all express the direction of the pole and location of the prime meridian of a body with respect to an inertial reference frame, as a function of time. This information defines the coordinate axes of the "Body Equator and Prime Meridian" system.

The orientation models use three Euler angles to describe the pole and prime meridian location: the first two angles, in order, are the right ascension and declination (henceforth ra and DEC) of the north pole of a body as a function of time. The third angle is the prime meridian location (represented by 'W'), which is expressed as a rotation about the north pole, also a function of time. The coordinate transformation defined by the Euler angles is represented by the matrix product

```
[ W ]    [ Pi/2 - Dec ]    [ Pi/2 + RA ]
     3                1                3
```

where

```
[ omega ]
         i
```

denotes the matrix that rotates vectors by –omega about the ith coordinate axis, using the right hand rule. (This notation is explained in detail in [214]).

In PCK files, the time arguments of functions that define orientation always refer to Barycentric Dynamical Time (TDB), measured in centuries or days past a specified epoch such as J2000, which is Julian ephemeris date 2451545.0. The time units expected by the SPICELIB software are ephemeris days for prime meridian motion and ephemeris centuries for motion of the pole.

NAIF Text Kernel Format

Text PCK files conform to a flexible format called "NAIF text kernel" format. (A number of SPICELIB kernels other than text PCKs use this NAIF text kernel format is described in detail in [218]. For the reader's convenience, an overview of the format is provided here.

NAIF text kernels are, first of all, ASCII files. As such, they are human readable and can be easily modified by text editors. In addition, NAIF text kernels can be readily ported between computer systems, even when the systems in question have different file systems and file formats.

The NAIF text kernel format provides for representation of data in a "keyword = value" syntax. The format also provides for the inclusion of free-form comment blocks.

There are two kinds of data that can be placed in NAIF text kernel files: double precision numbers and UTC time strings.

According to the text kernel format, a text kernel nominally consists of a series of sets of contiguous lines (or "blocks") of comments, alternating with blocks of data. Comment blocks are started with the string (called a "control sequence")

```
\begintext
```

alone on a line, as shown here. Comment blocks are ended by the control sequence

```
\begindata
```

alone on a line. In a text kernel file, the lines preceding the first

```
\begindata
```

control sequence are considered to constitute a comment block; the

```
\begintext control sequence is optional for this comment
```

block.

Comment blocks can contain arbitrary text, except for non-printing characters or lines that can be interpreted as control sequences. On the other hand, data must be organized according to a very specific format: all of the data in a text kernel must appear in the form of an "assignment" such as

```
NAME = ( VALUE1, VALUE2, ... )
```

where name is a string no longer than 32 characters, and the values on the right hand side are double precision numbers. A specific example is shown below:

```
BODY399_RADII     = (  6378.140  6378.140  6356.75  )
```

Some variations on the form shown here are allowed: commas between data values are optional, the right hand side of the assignment can be continued over multiple lines, and the numbers can be expressed as integers or reals without causing the PCK software to fail. Assignments of scalars do not require the value on the right hand side to be enclosed in parentheses, but that notation is frequently used as a visual cue. Blank lines within or between assignments are ignored by the SPICELIB software that reads text kernels.

In addition to numbers, UTC strings can be assigned to variables. The "@" character is used to identify the strings as time strings. The strings are stored internally as double precision numbers representing "UTC seconds past J2000." An example is the assignment:

```
SCLK_KERNEL_ID           = ( @01-MAY-1991/16:25 )
```

See [218] for a complete discussion of the allowed form of assignments.

The effect of an assignment in a text PCK file is to associate values with a name. The name is referred to as a "kernel variable." When a text PCK file is loaded by an application, the associations of names and values established by the PCK are maintained: the values associated with a given name can be retrieved at any time.

Text PCK Contents

Other than the limitations imposed by the PCK kernel formats, there are no absolute restrictions on the names or values of the variables used in PCK files. However, the SPICELIB kernel concept calls for the contents of PCK files to be limited to physical and cartographic constants describing extended solar system bodies: radii of bodies, constants defining orientation models, masses or values of GM are examples of data appropriate for inclusion in PCKs.

SPICELIB provides a text PCK access routine (RTPOOL) that can return the values of any variable defined in a text PCK file. There are also more specialized PCK access routines (BODEUL, BODMAT, TIPBOD, TISBOD, BODVAR) that can recognize and make use of a specific set of of kernel variables. These access routines return Euler angles and coordinate and state transformation matrices that are used to convert position and state vectors from inertial to body-fixed, planetocentric coordinates. In this document, the formulas defining time-varying coordinate transformation matrices and Euler angles are referred to as "orientation models" since they define the orientation of an extended body with respect to specific inertial frames.

Because PCK access routines that deal with orientation models are used extensively in SPICELIB and applications that use the Toolkit, the kernel variables that these routines rely on will be discussed in detail.

The functions that define the Euler angles are characterized by a set of parameters. The specific values of the parameters are values assigned to kernel variables in PCK files. The functions themselves are implemented by code within SPICELIB routines. The general form of the functions is that used in the latest IAU/IAG/COSPAR report. Values shown in this document reflect the 1991 report, [306]. For the latest PCK values, check with NAIF.

In a text PCK file, the variables (Euler angles)

```
RA,  DEC,  W
```

for the Earth are represented by the names

```
BODY399_POLE_RA
BODY399_POLE_DEC
BODY399_POLE_PM
```

The equations above are expressed in a text PCK file by the kernel variable assignments (Values taken from [306].)

```
BODY399_POLE_RA        = (     0.       -0.641         0. )
BODY399_POLE_DEC       = (   +90.       -0.557         0. )
BODY399_PM             = (  190.16  +360.9856235       0. )
```

Note that the string "PM" is used in place of "W" for the kernel variable associated with the prime meridian.

If you examine a PCK file produced by NAIF, you'll see an additional symbol grouped with the ones listed here; it is

```
BODY399_LONG_AXIS
```

This term is currently zero for all bodies except Mars. It represents the offset between the longest axis of the triaxial ellipsoid used to model a body and the prime meridian of the body.

Text PCK Kernel Variable Names

Text PCK kernel variables recognized by SPICELIB PCK access routines have names that follow a simple pattern: variables related to a body whose NAIF integer code is nnn have names of the form

```
BODYnnn_<item name>
```

where

```
<item name>
```

is a short string that identifies the type of quantity the kernel variable represents. For example, the variable containing quadratic polynomial coefficients for the right ascension of the Earth's north pole is

```
BODY399_POLE_RA
```

Note the number "399" appearing in the kernel variable names; this is the NAIF integer code for the Earth.

The specific item names that are recognized by PCK access routines are specified in the following sections.

Restrictions on the Form of Orientation Models in Text PCK Kernels

Orientation models that are usable by SPICE's text PCK access routines are not available for all solar system bodies. For example, Saturn's moon Hyperion is "tumbling" and does not admit a description of its motion by the sort of models used in text PCK kernels.

To date, no PCK files containing models for the rotation of comets have been produced or collected by NAIF. Models for the rotation of comets, as well as for the rotation of asteroids and the natural satellites of planets, can be accommodated by the PCK system only if the models have the form described in the section "Models for the Sun, Planets, and Asteroids" or in "Models for Satellites" or if binary PCK data is available.

Models for the Sun, Planets, and Asteroids in Text PCK Kernels

For the Sun, planets, and asteroids, the expressions used in text PCK files for the north pole direction and prime meridian location are always quadratic polynomials, where the independent variable is time. Some coefficients may be zero.

Let RA and DEC represent the right ascension and declination of a body's north pole, and let W be the prime meridian location, measured in the counterclockwise direction, from the cross product of the Earth's "mean" North pole at the J2000 epoch (as used in defining the J2000 frame) and BODY's North pole at et, to BODY's prime meridian at et.

The variables RA, DEC, and W constitute sufficient information to compute the transformation from a specified inertial frame to body-fixed, planetocentric coordinates for the body to which they apply, at a specified time.

The angles RA, DEC, and W are defined as follows:

```
                                 2
                            RA2*t
RA  =  RA0  +  RA1*t/T   +  ------
                               2
                              T

                                  2
                            DEC2*t
DEC =  DEC0 + DEC1*t/T   +  -------
                               2
                              T
```

```
                                      2
                                  W2*t
     W   =  W0   + W1*t/d    +   -----
                                     2
                                    d
```

where

```
     d = seconds/day
     T = seconds/Julian century
     t = ephemeris time, expressed as seconds past the reference epoch
         for this body or planetary system
```

Below is an example showing actual values of the parameters used to define the latest IAU model for orientation of the Earth. Here, the motions of both the pole and the prime meridian are given by linear polynomials; the quadratic terms are all zero. The values shown here are taken from [306]. In that document, the names

```
     alpha
          0
```

and

```
     delta
          0
```

are used in place of the names RA and DEC used here.

```
     alpha    =   0.00 - 0.641 T
          0

     delta    =  90.0  - 0.557 T
          0

     W        =  190.16 + 360.9856235 d

     T represents centuries past J2000 (TDB),

     d represents days past J2000 (TDB).
```

Models for Satellites in Text PCK Kernels

Orientation models for natural satellites of planets are a little more complicated; in addition to polynomial terms, the RA, DEC, and W expressions include trigonometric terms. The arguments of the trigonometric terms are linear polynomials. These arguments are usually called "phase angles." However, within SPICELIB internal documentation, these quantities are called "nutation precession angles"; for consistency with the SPICELIB software, we'll use this terminology in the following discussion.

Expressions for the right ascension and declination of the north pole and the location of the prime meridian for any satellite of a given planet are as follows:

$$RA = RA0 + RA1*t/T + \frac{RA2*t^2}{T^2} + \sum_i a_i \sin theta_i$$

$$DEC = DEC0 + DEC1*t/T + \frac{DEC2*t^2}{T^2} + \sum_i d_i \cos theta_i$$

$$W = W0 + W1*t/d + \frac{W2*t^2}{d^2} + \sum_i w_i \sin theta_i$$

where

```
d = seconds/day
T = seconds/Julian century
t = ephemeris time, expressed as seconds past a reference epoch
```

RA0, RA1, DEC0, DEC1, W0, and W1 are constants specific to each satellite.

The nutation precession angles

$$theta_i$$

are specific to each planet. The coefficients

$$a_i, \quad d_i, \quad \text{and } w_i$$

are specific to each satellite.

As an example, the nutation precession angles for Earth given by [252] are shown below. That document uses the variable names E1—E5 in place of

```
theta  --- theta
     1          5
```

in the equations above.

```
E1 = 125.045 -  0.052992 d
E2 = 250.090 -  0.105984 d
E3 = 260.008 + 13.012001 d
E4 = 176.625 + 13.340716 d
E5 = 357.529 +  0.985600 d

Here d represents days past J2000 (TDB)
```

Because the NAIF Toolkit software expects the time units for the angles to be centuries (as in the IAU models for most bodies—the Earth is an exception), the linear coefficients are scaled by 36525.0 for use in the kernel variable assignments:

```
BODY3_NUT_PREC_ANGLES  = (  125.045     -1935.5328
                            250.090     -3871.0656
                            260.008    475263.3
                            176.625    487269.6519
                            357.529     35999.04     )
```

In this assignment, the constant and linear polynomial coefficients for each nutation precession angle are listed together: the numbers 125.045 and -1935.5328 are the constant and linear terms for the angle "E1," and so on.

Note the body number 3 in the kernel variable name above; this number designates the Earth-Moon barycenter. The PCK access routines expect nutation precession angles to be associated with the barycenters of planetary systems.

Here are the expressions for the right ascension and declination of the moon, also taken from [252]:

```
alpha   =  270.000 +  0.003 T -  3.878 sin(E1)  -  0.120 sin(E2)
     0
                       +  0.070 sin(E3)  -  0.017 sin(E4)


delta   =  66.541  + 0.013 T  +  1.543 cos(E1)  +  0.024 cos(E2)
     0
                    -  0.028 cos(E3)  +  0.007 cos(E4)


W       =  38.317   + 13.1763582 d    +  3.558 sin(E1)
```

```
                          +  0.121 sin(E2)

                          -  0.064 sin(E3)

                          +  0.016 sin(E4)

                          +  0.025 sin(E5)

Here d represents days past J2000 (TDB),
and T represents Julian centuries past J2000 (TDB).
E1--E5 are the nutation precession angles.
```

The polynomial terms are assigned to symbols by the statements

```
BODY301_POLE_RA       = (  270.000    0.003       0. )
BODY301_POLE_DEC      = (  +66.541    0.013       0. )
BODY301_PM            = (   38.317  +13.1763582   0. )
```

The coefficients of the trigonometric terms are assigned to symbols by the statements

```
BODY301_NUT_PREC_RA  = (  -3.878  -0.120  +0.070  -0.017   0.    )
BODY301_NUT_PREC_DEC = (  +1.543  +0.024  -0.028  +0.007   0.    )
BODY301_NUT_PREC_PM  = (  +3.558  +0.121  -0.064  +0.016  +0.025 )
```

Note that for the RA and PM (prime meridian) assignments, the ith term is the coefficient of sin(Ei) in the expression used in the IAU model, while for the DEC assignment, the ith term is the coefficient of cos(Ei) in the expression used in the IAU model.

SPICELIB software for text PCK kernels expects the models for satellite orientation to follow the form of the model shown here: the polynomial terms in the RA, DEC, and W expressions are expected to be quadratic, the trigonometric terms for RA and W (satellite prime meridian) are expected to be sums of sines of nutation precession angles, and the trigonometric terms for DEC are expected to be sums of cosines of nutation precession angles. The nutation precession angles themselves are defined by linear polynomial functions of time.

Note that the number of values defining the nutation precession angles for a planetary system must be be consistent with the number of trigonometric terms used in the expressions for the RA, DEC and W angles for the satellites of that system. See “Creating and Modifying Text PCKs Kernels” for details.

Epoch and Frame Specifications in Text PCK Kernels

The constants used in PCK files to define an orientation model for a specified body are assumed by default to define a time-dependent rotation R(t) that converts vectors from J2000 coordinates to body-fixed, planetocentric coordinates at the epoch t seconds past J2000, TDB (JED 2451545.0). We say that the constants are "referenced to the J2000 epoch and J2000 frame." However, these default values for the epoch and frame of the constants may be overridden: it is possible to use constants that are referenced to the B1950 frame and to the J1950 epoch, for example.

The default constants and frame for a body are overridden by setting the values of the kernel variables

```
BODY<id code>_CONSTANTS_REF_FRAME
```

and

```
BODY<id code>_CONSTANTS_JED_EPOCH
```

Here

```
<id code>
```

is:

- for planets and their satellites: the NAIF integer code of the corresponding planetary system's barycenter.
- for other bodies: the NAIF integer code of the body itself.

The values of the frame specifier variable

```
BODY<id code>_CONSTANTS_REF_FRAME
```

are the codes returned by the SPICELIB routine IRFNUM. Some of the commonly used codes and the corresponding frames are:

```
Code        Frame
----        -----
  1         J2000

  2         B1950

  3         FK4 (used in CSPICE for 'EME50')
```

For example, to use constants referenced to the FK4 or EME50 frame for the asteroid Gaspra (ID code = 9511010), the PCK file containing the constants should include the assignment

```
BODY9511010_CONSTANTS_REF_FRAME   =   (  3  )
```

The values of the epoch specifier variable

```
BODY<id code>_CONSTANTS_JED_EPOCH
```

are Julian ephemeris dates. To use constants for Gaspra that are referenced to the J1950 epoch, the PCK file containing the constants should include the assignment

```
BODY9511010_CONSTANTS_JED_EPOCH   =   (  2433282.5D0 )
```

The creator of a PCK file can set the frame and epoch of the constants on a body-by-body basis, except in the case of planets and their (natural) satellites, where a single choice of frame and epoch must be used for each planetary system. For example, to use constants referenced to the B1950 frame and J1950 epoch for the Earth and Moon, you would make the assignments

```
BODY3_CONSTANTS_REF_FRAME   =   (  2           )
BODY3_CONSTANTS_JED_EPOCH   =   (  2433282.5D0 )
```

The ID code '3' designates the Earth-Moon barycenter.

Note: the assignment

```
BODY399_CONSTANTS_REF_FRAME   =   (  2           )
BODY399_CONSTANTS_JED_EPOCH   =   (  2433282.5D0 )
```

would be ignored by the PCK reader routines; you cannot assign a frame or epoch using the ID code of a planet or satellite.

Using constants that are referenced to frames or epochs other than J2000 does not affect the definitions of the inputs or outputs of any of the PCK access routines. These routines make the necessary transformations internally to take into account the reference frame and epoch of the orientation constants in the kernel pool. Thus the PCK access routine BODMAT, which has the argument list

```
BODMAT ( BODY, ET, TIPM )
```

returns the J2000-to-body-fixed transformation matrix for ephemeris time expressed as "ET" seconds past the J2000 epoch (JED 2451545.0), regardless of the epoch or frame of the orientation constants used to compute TIPM.

Shape models in Text PCK Kernels

SPICELIB contains a number of geometry routines that make use of triaxial ellipsoidal models of extended solar system bodies. Although SPICELIB currently contains no routines that directly use the specific PCK variables that define these models, text PCK files typically contain radii of solar system bodies, since these values can be looked up by low-level text PCK access routines and subsequently used by SPICELIB geometry routines.

In text PCK files produced by NAIF, the radius values for body nnn are assigned to the variable

```
BODYnnn_RADII
```

Three radius values are always assigned for each instance of this variable. The data are ordered as in [306]: the equatorial radii are listed with the largest axis, often called the "a" axis, appearing first; the polar radius is last. Spheroids and spheres are obtained when two or all three radii are equal.

Example: Radii of the Earth.

```
BODY399_RADII     = (     6378.140    6378.140     6356.75    )
```

Summary of PCK Variables used in Text PCK Kernels by SPICE

In order to compute transformations for the Sun, a planet, or an asteroid (say body number ppp), the PCK access routines require that one or more PCK files containing values for the following variables be loaded:

```
BODYppp_POLE_RA
BODYppp_POLE_DEC
BODYppp_PM
```

For a satellite (say body number sss), one or more PCK files containing values for the following variables must be loaded:

```
BODYsss_POLE_RA
BODYsss_POLE_DEC
BODYsss_PM
BODYsss_NUT_PREC_RA
BODYsss_NUT_PREC_DEC
BODYsss_NUT_PREC_PM
BODYbbb_NUT_PREC_ANGLES
```

where the code bbb embedded in the last name above is that of the barycenter of the planetary system to which the satellite belongs.

The triaxial ellipsoidal model for body nnn is expressed by the assignment

```
BODYnnn_RADII = ( <larger equatorial radius>,
                  <smaller  equatorial radius>,
                  <polar radius> )
```

Creating and Modifying Text PCKs

The PCK file text format allows NAIF Toolkit users to easily modify existing text PCKs and to create their own files containing values of their choosing. Any text editor capable of working with ASCII files can be used to edit text PCK files.

Although the text PCK format makes it easy to modify text PCK files, NAIF recommends that application programmers avoid software designs that call for special-purpose, user-created text PCK files. The opportunities for confusion and errors increase with the number of available versions of a text PCK file (or any data file).

NAIF recommends that you take the following precautions when modifying a text PCK file:

- Change the name of the updated file.
- Document the changes by adding appropriate comments to the file. Each text PCK file should contain sufficient information to allow a human reader to find out who was responsible for creating the current version of the file and what the source was for each data value in the file. If the file is an update, the reason for the update and a summary of the differences from the previous version should be included.
- Test the file using software that makes use of any values that you've added or modified.

The reasons why a NAIF Toolkit user might wish to modify an existing text PCK are:

- Removing unneeded data or comments to speed up loading and simplify the file. Removal of data is much more important than removal of comments, as far as speeding up kernel loading is concerned.
- Adding data values for new bodies.
- Updating existing data values or substituting preferred data values.

New kernel variables added to text PCK files should follow the naming conventions described in the "Kernel Variable Names" section. All text PCK variable names, whether or not they are recognized by SPICELIB software, should start with the prefix

```
BODYnnn_
```

where nnn is the NAIF integer code of the body the variable applies to.

Kernel variables having names recognized by users' application software are a potential problem area: if the names used in the application don't match those in the text PCK file, the application will fail to obtain the data as intended. The most frequent cause of this type of failure is misspelling of variable names, but programmers who are considering changing names of PCK variables that are already in use should also keep this problem in mind.

Modifying orientation models for satellites requires attention to the consistency between the number of nutation precession angles and the number of coefficients of trigonometric functions having the nutation precession angles as arguments. For any planetary system, there should be twice as many values for the nutation precession angles as the maximum number of trigonometric terms in the expressions for prime meridian location or right ascension or declination of the pole of any satellite in the system. This is because the nutation precession angles are defined by linear polynomials; each polynomial has two defining coefficients.

For example, the 1991 IAU model for the Earth-Moon system uses the following values to determine the Moon's orientation:

```
BODY3_NUT_PREC_ANGLES  = (  125.045    -1935.5328
                            250.090    -3871.0656
                            260.008   475263.3
                            176.625   487269.6519
                            357.529    35999.04      )

BODY301_POLE_RA        = (  270.000    0.003         0. )
BODY301_POLE_DEC       = (  +66.541    0.013         0. )
BODY301_PM             = (   38.317  +13.1763582     0. )

BODY301_NUT_PREC_RA  = (  -3.878  -0.120  +0.070  -0.017   0.     )
BODY301_NUT_PREC_DEC = (  +1.543  +0.024  -0.028  +0.007   0.     )
BODY301_NUT_PREC_PM  = (  +3.558  +0.121  -0.064  +0.016  +0.025 )
```

Note that there are five nutation precession angles, with two coefficients defining each one. The trigonometric terms in the expressions for the right ascension, declination, and prime meridian location for the moon require five nutation precession angles.

If a NAIF Toolkit user were to update this model with a new one that had, for example, only one nutation precession angle, then the variables

```
BODY301_NUT_PREC_RA
BODY301_NUT_PREC_DEC
BODY301_NUT_PREC_PM
```

would each have to be assigned only one value. Failure to make this change would result in an error being signalled at run-time by any SPICELIB PCK access routine that uses these variables.

Binary PCK Kernel Format

Segments–The Fundamental PCK Building Blocks

A binary PCK file contains one or more 'segments'. Each segment contains data sufficient to compute the axes of a body-fixed planetary coordinate system, relative to a specified inertial reference frame, as a function of time.

The data in each segment are stored as a single array. The summary for the array, called a 'descriptor', has two double precision components:

1. The initial epoch of the interval for which data are contained in the segment, in ephemeris seconds past Julian year 2000;
2. The final epoch of the interval for which data are contained in the segment, in ephemeris seconds past Julian year 2000.

The descriptor has five integer components:

1. The NAIF integer code for the body;
2. The NAIF integer code for the inertial reference frame;
3. The integer code for the representation (type of PCK data). Only type 2 is presently supported;
4. The initial address of the array;
5. The final address of the array.

The name of each array may contain up to 40 characters. This space may be used to store a 'pedigree' for the data in the array. The pedigree of a segment should allow a user to determine the conditions under which the data in the segment were generated.

The Comment Area

Preceding the 'segments', the Comment Area provides space in a binary PCK file for storing additional textual information besides what is written in the array names. Ideally, each binary PCK file would contain internal documentation that describes the origin, recommended use, and any other pertinent information about the data in that file. For example, the beginning and ending epochs for the file, the names and NAIF integer codes of the bodies included, an accuracy estimate, the date the file was produced, and the names of the source files used in making the binary PCK file could be included in the Comment Area.

SPICELIB provides a family of subroutines for handling this Comment Area. This software provides the ability to add, extract, read, and delete comments and convert commented files from binary format to transfer format and back to binary again.

Binary PCK Data Types

The third integer component of the descriptor—the code for the representation, or 'data type'—is the key to the binary PCK format. For purposes of determining the segment best suited to fulfill a particular request, all segments are treated equally. It is only when the data in a segment are to be evaluated that the type of data used to represent the data becomes important. Because this step is isolated within low-level readers, new data types can be added to the binary PCK format without affecting application programs that use the higher level readers.

Supported Data Types

Two representations, or data types, are currently supported by the binary PCK routines in SPICELIB. They are Chebyshev polynomials (Euler angles only), called "Type 2" and Chebyshev polynomials (Euler angles and their derivatives) for intervals of possibly varying lenghts, called "Type 3".

Type 2: Chebyshev (Angles only)

These are sets of Chebyshev polynomial coefficients for the Euler angles, defining right ascension and declination of the body's north pole, and its prime meridian rotation as a function of time. The rates of the angles are obtained by differentiation.

Each segment contains an arbitrary number of logical records. Each record contains a set of Chebyshev coefficients valid throughout an interval of fixed length. The subroutine PCKE02 contains the algorithm used to construct a set of Euler angles from a particular record and epoch.

The records within a segment are ordered by increasing initial epoch. All records contain the same number of coefficients. A segment of this type is structured as follows:

```
+---------------+
 Record 1               +---------------+
 Record 2               +---------------+
   .
   .
   .
+---------------+
 Record N               +---------------+
 INIT                   +---------------+
 INTLEN                 +---------------+
 RSIZE                  +---------------+
 N                      +---------------+
```

A four-number 'directory' at the end of the segment contains the information needed to determine the location of the record corresponding to a particular epoch.

1. INIT is the initial epoch of the first record, given in ephemeris seconds past 2000 Jan 01 12:00:00, also known as J2000.
2. INTLEN is the length of the interval covered by each record, in seconds.
3. RSIZE is the total size of (number of array elements in) each record.
4. N is the number of records contained in the segment.

Each record is structured as follows:

```
+------------------+
 MID                      +------------------+
 RADIUS                   +------------------+
 X  coefficients          +------------------+
 Y  coefficients          +------------------+
```

```
                 Z  coefficients          +------------------+
```

The first two elements in the record, MID and RADIUS, are the midpoint and radius of the time interval covered by coefficients in the record. These are used as parameters to perform transformations between the domain of the record (from MID - RADIUS to MID + RADIUS) and the domain of Chebyshev polynomials (from -1 to 1).

The same number of coefficients is always used for each component, and all records are the same size (RSIZE), so the degree of each polynomial is

(RSIZE - 2) / 3 - 1

Type 3: Chebyshev (Angles and derivatives of intervals of possibly varying lengths)

These are sets of Chebyshev polynomial coefficients for the Euler angles, defining right ascension (RA) and declination (DEC) of the body's north pole, and its prime meridian rotation (W) as a function of time, and their derivatives. This data type was created for the attitude of the asteroid Eros.

Each segment contains an arbitrary number of logical records. Each record contains a set of Chebyshev coefficients for angles and derivatives valid throughout an interval defined in the record. The lengths of these intervals may vary within a segment. The subroutine PCKE03 contains the algorithm used to construct a set of Euler angles from a particular record and epoch. All records contain the same number of coefficients.

A segment of this type is structured as follows:

```
   ----------------------------------------------------
   | Record 1 | Record 2 | ... | Record N-1 | Record N |
   ----------------------------------------------------
```

where each record has the following format:

```
   --------------------------------------------------
   |  Midpoint of the approximation interval in TDB  |
   --------------------------------------------------
   | Radius of the approximation interval in seconds |
   --------------------------------------------------
   |        coefficients for the RA position         |
   --------------------------------------------------
   |        coefficients for the DEC position        |
   --------------------------------------------------
   |        coefficients for the W position          |
   --------------------------------------------------
```

```
|        coefficients for the RA velocity        |
--------------------------------------------------
|        coefficients for the DEC velocity       |
--------------------------------------------------
|        coefficients for the W velocity         |
--------------------------------------------------
```

TDB is time in ephemeris seconds past J2000, called et in the SPICELIB system.

Creating Binary PCKs

Only very knowledgeable users who need to incorporate new planetary/satellite orientation information in binary format should consider writing binary PCK files. Users who write binary PCK files must have a thorough understanding of the information they wish to place in a binary PCK file. They must also master the high level structure of the PCK files, and they must be sure to correctly package the data for the PCK writing subroutines provided in SPICE. We also strongly recommend that the writer of a PCK file include descriptive comments in the comment area. Normally, binary PCK files should be obtained from NAIF.

To create files that are smaller and faster to load, the user should keep in mind that the PCK segments should be as large as possible.

The are generally three steps to creating a binary PCK file.

1. Open the file.
2. Begin the segment, add data to the segment and close the segment.
3. Close the file.

The subroutine PCKOPN is used to open a new binary PCK file. Below is an example for a call to PCKOPN. 'NAME' is the name of the file to be opened, IFNAME is the internal file name, HANDLE is the handle of the opened SPK file. We use I for the number of record to reserve for comments.

```
CALL PCKOPN ( FILE, IFNAME, I, HANDLE )
```

The method for beginning the segment, adding data to the segment and closing the segment differs with the PCK type.

For segments of type 2, there is a segment writing routine called PCKW02 in SPICELIB. This routine takes as input arguments the handle of an PCK file that is open for writing, the information needed to construct the segment descriptor, and the data to be stored in the segment. The header of the subroutine provides a complete description of the input arguments and an example of its usage. Here's an example of a call to PCKW02:

```
CALL PCKW02 ( HANDLE,  BODY,  FRAME, FIRST,
                LAST,  SEGID, INTLEN,     N,
              POLYDG, CDATA,  BTIME          )
```

For type 3, there are three subroutines used in creating a binary PCK file. They are PCK03B, which begins a type 3 segment, PCK03A, which adds data to segment, and PCK03E, which ends a segment. The type 3 subroutines can be used in a loop, where PCK03A is called to add data to the segment. Here is a code fragment which begins a type 3 segment, writes data to that segment in a loop, and then closes the segment.

```
CALL PCK03B ( HANDLE, SEGID, BODY, FRAME,
              ETSTRT, ETSTOP, CHBDEG )
DO WHILE ( <a condition> )
   ...
   CALL PCK03A ( HANDLE, N, COEFFS, EPOCHS )
   ...
END DO

CALL PCK03E ( HANDLE )
```

When a user is finished writing segments to an PCK file, the file must be closed with the subroutine PCKCLS.

```
CALL PCKCLS ( HANDLE )
```

PCK Software

This section describes the specifications and proper use of the SPICELIB PCK software.

Getting PCK Data into Your Program

Because loading PCK files is usually time-consuming, it is good programming practice to have applications load PCK files during program initialization rather than throughout their main processing thread, especially if that processing thread is a loop.

It is also wise to avoid designing data processing systems that effectively place PCK loading in a tight loop by requiring repeated runs of programs that expend a significant fraction of their run time on loading PCK files. If a program loads PCK files, it is preferable that it do all of its processing in a single run, or at least in a small number of runs, rather than carry out its processing by being re-run a large number of times: the former design will greatly reduce the ratio of the time the program spends loading PCKs to the time it spends on the rest of its data processing.

Loading Text PCK Kernels

As mentioned in the "System Overview" section, in order to use text PCK files in an application, the data in the files must be read into memory. This is accomplished by calling the SPICELIB routine LDPOOL. The name of the text PCK file to load is supplied as an input to LDPOOL, for example:

```
CALL LDPOOL ( 'P_CONSTANTS.KER' )
```

File names supplied to LDPOOL will generally be system-dependent. It is good programming practice to not use hard-coded file names in calls to LDPOOL. Instead, applications should obtain kernel file names by one of the following methods:

- Reading the kernel file names from a file containing the names. (This allows users to change the kernel files without re-compiling and re-linking the application.)
- Prompting the user for the file names at run-time.

An application can load any number of text PCK files during a single program run. There are, however, parameterized limits on both the total number of kernel variables that can be stored and on the total number of data values assigned to those variables.

Each time a text PCK is loaded, the assignments made in the file are maintained in the PCK software. In particular, if a kernel variable occurs in multiple PCKs that are loaded in a single run of a program, the value of the variable will be the one assigned in the following priority: last binary PCK file loaded, previously loaded binary PCK files, then last loaded text PCK files followed by previously loaded text PCK files. All binary PCK files take precedence over text PCK files. Within the binary and/or text file groups, the last loaded files takes precedence.

Loading Binary PCK Kernels

The routine PCKLOF maintains a database of loaded binary PCK files. The calling program indicates which files are to be used by passing their names to PCKLOF.

```
CALL PCKLOF ( 'example.pck', HANDLE)
```

PCKLOF returns a DAF file handle for each file, which may be used to access the file directly using DAF subroutines. Once an PCK file has been loaded, it may be accessed by the PCK software. Each set of constants is computed from a distinct segment.

An PCK file may contain any number of segments. In fact, the same file may contain overlapping segments: segments containing data for the same body over a common interval. When this happens, the latest segment in a file supersedes any competing segments earlier in the file. Similarly, the latest file loaded supersedes any earlier files. In effect, several loaded files become equivalent to one large file. Binary PCK files take precedence over text PCK files.

Unloading Binary PCK Kernels

It is possible, though unlikely, that a program would need to make use of many binary PCK files in the course of a single execution. On the other hand, the number of binary PCK files that may be open at any one time is limited, so such a program might need to unload some PCK files to make room for others. A binary PCK file may be unloaded by supplying its handle to subroutine PCKUOF. The call to this subroutine is shown below,

```
CALL PCKUOF ( HANDLE )     { Unload binary PCK file }
```

High-Level Access Routines

SPICELIB contains two basic categories of PCK access routines: those that return PCK data directly, and those that return quantities derived from PCK data. This section discusses the PCK access routines in the later category: these routines deal with coordinate and state transformations.

All of the routines listed here make use of the orientation models discussed in the section titled "Orientation Models used by PCK Software." Note that in order to use these routines, an application must first load a PCK file (or files) containing sufficient data to define all of the required orientation models. If needed data has not been loaded, these routines will signal run-time errors when called.

To obtain the matrix that transforms vectors from a specified inertial reference frame to body-fixed planetocentric coordinates for a specified body, at a specified ephemeris time, use the routine TIPBOD. The calling sequence is

```
CALL TIPBOD ( REF, BODY, ET, TIPM )
```

In the argument list for TIPBOD:

`REF` is a character string that designates an inertial reference frame recognized by SPICELIB

`BODY` is the NAIF integer code of the body defining the planetocentric coordinate system

`ET` is the ephemeris time at which the orientation model given the basis vectors of the planetocentric frame is to be evaluated

`TIPM` is the 3x3 transformation matrix that carries out the desired transformation.

For example, let V50 be a vector specified relative to the B1950 inertial reference frame. The call

```
CALL TIPBOD ( 'B1950', 399, ET, TIPM )
```

returns a matrix TIPM such that left-multiplying V50 by TIPM— accomplished in SPICELIB by the call

```
CALL MXM ( TIPM, V50, VBODFX )
```

will return the vector VBODFX specified relative to the Earth's ("body-fixed") planetocentric frame at time ET.

Note that many practical applications of TIPBOD require that a light-time corrected value of et be supplied as an input. See "Computing the Sub-Observer Point" in the "Examples" section below.

The subroutine BODMAT is an older, less general version of TIPBOD that does not accept an input argument specifying an inertial reference frame. Instead, BODMAT assumes that the frame is J2000. The calling sequence is the same as that of TIPBOD, minus the argument ref:

```
CALL BODMAT ( BODY, ET, TIPM )
```

The fundamental quantities defined by PCK orientation models are actually Euler angles, not matrices. These Euler angles, which we call "RA, DEC, and W," are related to the matrix TIPM shown above by the equation

```
TIPM = [ W ]   [ Pi/2 - DEC ]   [ Pi/2 + RA ]
            3                 1                3
```

The units of these angles are radians. To retrieve these angles directly, the call

```
BODEUL ( BODY, ET, RA, DEC, W, LAMBDA )
```

can be used. The arguments BODY and ET have the same meanings as in the argument lists of TIPBOD and BODMAT. The arguments RA and DEC represent the right ascension and declination of the North pole of body at et with respect to the J2000 inertial reference frame. The argument W is the prime meridian location for BODY at ET, also measured with respect to the J2000 inertial reference frame.

The output LAMBDA is the positive west longitude, measured from the prime meridian of body, of the longest axis of the triaxial ellipsoidal model for body given in a PCK file.

Currently, the only body having a non-zero value of LAMBDA is Mars (see [195]). SPICELIB software does not currently make use of LAMBDA.

SPICELIB provides a routine analogous to TIPBOD for transforming state vectors from inertial to planetocentric coordinates. This routine is called TISBOD; the calling sequence is

```
CALL TISBOD ( REF, BODY, ET, TSIPM )
```

The input arguments REF, BODY, and ET have the same meanings as in the argument list of TIPBOD. The output argument TSIPM is the 6x6 matrix required to transform state vectors from inertial to body-fixed coordinates.

Left multiplication of a state vector by TSIPM will transform it from the frame specified by REF to body-fixed planetocentric coordinates for BODY at time ET.

See "Transforming a State to Body-Fixed Coordinates" in the "Examples" section for further discussion of use of TISBOD.

Low-Level Access to Text PCK Kernels

WARNING: These low-level access routines for text PCK files only search the text kernel pool for these values. Values found in loaded binary PCK files will NOT be found by these routines. The values retrieved from a binary PCK file take precedence over the values found in the text PCK kernels. Therefore, if binary kernels have been loaded values returned by these low level routines may not be the same values used by higher level routines like TISBOD and TIPBOD. We recommend the user who loads binary PCK kernels NOT USE these low-level routines!

The lowest-level SPICELIB PCK access routine is RTPOOL. RTPOOL is a general-purpose routine for retrieving any text kernel data loaded via LDPOOL. To obtain data from RTPOOL, use the calling sequence

```
CALL RTPOOL ( NAME, N, VALUES, FOUND )
```

The meanings of the arguments of RTPOOL are follows:

`NAME` is the name of the kernel variable whose values are desired. This is the name used in a PCK file to make an assignment.

`N` is the number of data values assigned to the kernel variable.

`VALUES` is a double precision array of sufficient length to contain all of the data assigned to the kernel variable. Caution: if VALUES is declared with too short a length, RTPOOL will likely overwrite memory in the routine where VALUES is declared.

`FOUND` is a logical flag indicating whether the kernel variable designated by name was actually loaded.

RTPOOL is frequently used by other SPICELIB routines; however, SPICELIB users will generally be able to use the more convenient PCK access routine BODVAR, which is described below.

In text PCK's produced by NAIF, PCK kernel variables will have names conforming to the naming convention used in SPICELIB, that is, the kernel variable names have the form

```
BODYnnn_<item name>
```

The SPICELIB routine BODVAR can be used in place of RTPOOL to retrieve values for such variables; BODVAR accepts as inputs the NAIF integer code of the body and a string making up the portion of the item's name following the prefix.

```
BODVAR ( BODY, ITEM, DIM, VALUES )
CALL BODVAR ( 399, 'RADII', DIM, VALUES )
```

which would return the dimension and values associated with the variable

```
BODY399_RADII
```

namely

```
DIM      = 3
VALUE(1) = 6378.140
VALUE(2) = 6378.140
VALUE(3) = 6356.755
```

It is possible to test whether a kernel variable has been loaded by calling the SPICELIB logical function BODFND, as long as the variables in question follow the SPICELIB naming convention. The calling sequence is

```
FOUND = BODFND ( BODY, ITEM )
```

where body is the NAIF integer code of the body, and ITEM is the string making up the portion of the item's name following the prefix

```
BODYnnn_
```

For example, to test whether values for Jupiter's radii have been loaded, the test

```
FOUND = BODFND ( 599, 'RADII' )
```

could be used. The variable FOUND would be returned as .TRUE. if a PCK file had been loaded containing an assignment of the variable

```
BODY599_RADII
```

Examples

This section illustrates some typical applications of SPICELIB PCK software.

Computing the Sub-Observer Point

In this example, we present a small program that solves a realistic science analysis geometry problem: finding the sub-observer point on an extended body at a specified time. We will assume that we have an SPK file available that contains ephemeris data for the observer and target body at the time of interest.

```
      PROGRAM SUBOBS
C
C     Compute the planetocentric latitude and longitude of the
C     sub-observer point on an extended target body at a
C     specified UTC epoch.  The observer may be a spacecraft,
C     planet, or any body for which ephemeris data is available
C     in an SPK file.
C
C     This example uses text PCK kernels.  This program could
C     be modified to use binary PCK kernels by replacing the
C     LDPOOL call with a call to PCKLOF.  Both types of PCK
C     kernels could be used if a call to PCKLOF were added to
C     this program.
C

C
C     SPICELIB functions
C
      DOUBLE PRECISION      DPR

C
C     Local parameters
C
      INTEGER               FILEN
      PARAMETER           ( FILEN = 128 )

C
C     Local variables
C
      CHARACTER*(FILEN)     LEAP
      CHARACTER*(FILEN)     PCK
      CHARACTER*(FILEN)     SPK
      CHARACTER*(30)        UTC
      CHARACTER*(5)         CORR

      DOUBLE PRECISION      ALT
      DOUBLE PRECISION      DIST
      DOUBLE PRECISION      EPOCH
```

```
      DOUBLE PRECISION      ET
      DOUBLE PRECISION      LT
      DOUBLE PRECISION      OBSPOS ( 3 )
      DOUBLE PRECISION      RADII  ( 3 )
      DOUBLE PRECISION      STATE  ( 6 )
      DOUBLE PRECISION      SUBLAT
      DOUBLE PRECISION      SUBLON
      DOUBLE PRECISION      SUBPT  ( 3 )
      DOUBLE PRECISION      TIPM   ( 3, 3 )

      INTEGER               HANDLE
      INTEGER               N
      INTEGER               OBSRVR
      INTEGER               TARGET

      LOGICAL               CONT

      DATA CONT / .TRUE. /

C
C     Start out by prompting for the names of kernel files.
C     Load each kernel as the name is supplied.
C
      WRITE (*,*) ' '
      WRITE (*,*) 'Enter name of SPK file'
      READ  (*,FMT='(A)') SPK

      CALL SPKLEF ( SPK, HANDLE )

      WRITE (*,*) 'Enter name of leapseconds kernel'
      READ  (*,FMT='(A)') LEAP

      CALL LDPOOL ( LEAP )

      WRITE (*,*) 'Enter name of text P_constants kernel'
      READ  (*,FMT='(A)') PCK

      CALL LDPOOL ( PCK )


      DO WHILE ( CONT )

         WRITE (*,*) ' '
         WRITE (*,*) 'Enter NAIF integer code of target body'
         READ   *,    TARGET

         WRITE (*,*) 'Enter NAIF integer code of observer body'
         READ   *,    OBSRVR

         WRITE (*,*) 'Enter UTC epoch of observation'
         READ  (*,FMT='(A)') UTC

         WRITE (*,*) 'Enter aberration correction to use:'
         WRITE (*,*) 'NONE, LT, or LT+S'
         READ  (*,FMT='(A)') CORR

C
C        Convert the UTC epoch to ET.
C
```

```
      CALL UTC2ET ( UTC, ET )

C
C     Find the state of the target as seen from the observer at
C     ET.
C
      CALL SPKEZ ( TARGET, ET, 'J2000', CORR, OBSRVR, STATE,
     .             LT                                        )

C
C     The pure mathematical problem embedded in this application
C     is finding the nearest point on a triaxial ellipsoid to a
C     specified point.  The SPICELIB routine NEARPT solves this
C     problem.  NEARPT deals with an ellipsoid that is centered
C     at the origin and whose axes are lined up with the x, y,
C     and z coordinate axes.  So to use NEARPT, we'll have to
C     convert the problem at hand into a form that NEARPT can
C     accept.
C
C     The way to do this is to convert all of our vectors into
C     body equator and prime meridian coordinates.  The body
C     equator and prime meridian system is one in which the
C     coordinate axes are lined up with the body's geometric
C     axes (there may be a few exceptions to the rule--check the
C     PCK file to be sure about the body you're working with).
C

C
C     Find the epoch for which the inertial to body equator and
C     prime meridian transformation should be computed.
C
      IF ( CORR .EQ. 'NONE' ) THEN
         EPOCH = ET
      ELSE
         EPOCH = ET - LT
      END IF

C
C     Find the transformation from inertial to body equator and
C     prime meridian coordinates at the appropriate epoch.
C
      CALL TIPBOD ( 'J2000', TARGET, EPOCH, TIPM )

C
C     Negate the observer-target vector and rotate the result
C     into body-fixed coordinates.
C
      CALL VMINUS ( STATE, OBSPOS )

      CALL MXV    ( TIPM,  OBSPOS,  OBSPOS )

C
C     Look up the radii of the target body from the PCK file.
C
      CALL BODVAR ( TARGET, 'RADII', N, RADII )

C
C     Find the nearest point on the body to the observer.
C
```

```
            CALL NEARPT ( OBSPOS, RADII(1), RADII(2), RADII(3),
     .                    SUBPT,  ALT                         )

C
C           Find the latitude and longitude of the sub-observer point.
C
            CALL RECLAT ( SUBPT,  DIST,  SUBLON,  SUBLAT )

C
C           Write out the results, converting radians to degrees.
C
            WRITE(*,*) ' '
            WRITE(*,*) 'Sub-observer longitude (deg) ', DPR() * SUBLON
            WRITE(*,*) 'Sub-observer latitude  (deg) ', DPR() * SUBLAT
            WRITE(*,*) ' '

            WRITE (*,*) 'Do you wish to continue? (T/F)'
            READ   *,    CONT

         END DO

         END
```

Transforming a State to Body-Fixed Coordinates

Occasionally, it may be useful to transform a state vector into body-fixed coordinates. To perform the transformation correctly, it is necessary to take into account the time derivative of the inertial-to-body-fixed coordinate transformation. A discussion of the error made by ignoring this derivative term is given in [214].

The following code fragment shows how to use the SPICELIB routine TISBOD to transform state vectors into body-fixed coordinates.

In the following code fragment, TISBOD is used to transform a state in J2000 inertial coordinates to a state in body-fixed coordinates.

The 6-vector STATE represents the inertial state (position and velocity) of an object with respect to the center of the body at time et. The matrix TSIPM is declared as a 6x6 double-precision matrix.

This program loads both a text and a binary PCK file. The question of from which file is the data retieved is answered by the coverage of the files. If a particular requested time and body is covered by the loaded binary file, the binary data will be used. If not, the text data will be used.

For example, say the binary file covers body 301 from 1985 through 1995. It the 'ORIENT_BODY.PCK' file has body 301 during this time period, that data will be used. Otherwise the program will use the data from 'PLANETARY_CONSTANTS.KER.'

```
C
C     First load the kernel pool.
C
      CALL LDPOOL ( 'PLANETARY_CONSTANTS.KER' )

C
C     We can also load a binary PCK file.  The software
C     will use values from the binary PCK, if available,
C     and the text PCK above, if not.
C
      CALL PCKLOF ( 'ORIENT_BODY.PCK', HANDLE )

C
C     Next get the inertial to body-fixed transformation
C     and its derivative.
C
      CALL TISBOD ( 'J2000', BODY, ET, TSIPM )

C
C     Convert position and velocity to body-fixed coordinates.
C
      CALL MXVG    ( TSIPM, STATE, 6, 6, BDSTAT )
```

Creating a type 03 PCK file.

We assume that we are creating a new file for this example. In addition, we will make use of the fictitious subroutine GENREC to generate a type 03 PCK data record. This is for demonstrative purposes only, and is not intended to be a complete program.

The following routines will be used in this example:

- PCKCLS – Close a PCK file.
- PCKOPN – Open a new PCK file.
- PCK03B – Begin a type 03 PCK segment.
- PCK03A – Add data to a type 03 PCK segment.
- PCK03E – End a type 03 PCK segment.

For the complete details of these routines, please see the appropriate source code files.

Variable declarations for the example code fragment.

```
CHARACTER*(128)       FILE
CHARACTER*(60)        IFNAME
CHARACTER*(32)        FRAME
CHARACTER*(40)        SEGID

DOUBLE PRECISION      ET
DOUBLE PRECISION      ETSTRT
DOUBLE PRECISION      ETSTOP
DOUBLE PRECISION      RECORD(*)
DOUBLE PRECISION      STEP

INTEGER               BODY
INTEGER               CHBDEG
INTEGER               HANDLE
```

```
C     THIS IS A CODE FRAGMENT, NOT A PROGRAM!
C
C     Assign the name of the file to be created.
C
      FILE   = 'new_pck.bpc'
C
C     Assign an internal filename for the file.
C
      IFNAME = 'THIS IS A NEW PCK FILE'
C
C     Open the PCK file.
C
      CALL PCKOPN ( FILE, IFNAME, 0, HANDLE )
C
C     Set the identifier for the segment. This is a character
```

```
C     string of at most 40 printing ASCII characters. It may
C     be blank.
C
      SEGID = 'This is a test. This is only a test.'
C
C     Set the body for the PCK segment. We will use the the
C     Moon, ID code 301.
C
      BODY   = 301
C
C     We will be using the J2000 reference frame for the segment.
C
      FRAME = 'J2000'
C
C     Set the degree of the Chebyshev polynomials used for the
C     segment.
C
      CHBDEG = 10
C
C     Begin the PCK segment.
C
      CALL PCK03B ( HANDLE, SEGID,  BODY,   FRAME,
     .                      ETSTRT, ETSTOP, CHBDEG )
C
C     We have a start time and a stop time available, ETSTRT
C     and ETSTOP, and we want to generate records for
C     sub-intervals of the interva (ETSTRT, ETSTOP). For
C     simplicity, we will assume equal width intervals and
C     that (ETSTOP-ETSTRT)/STEP is a positive integer. The
C     times are in ET, ephemeris seconds past J2000.
C
C     The time step will be one day, 86400 seconds.
C
      STEP = 86400.0D0
C
C     Set the initial time.
C
      ET  = ETSTRT
C
C     Loop until we have processed all of the time intervals,
c     i.e., when ET >= ETSTOP.
C
      DO WHILE ( ET .LT. ETSTOP )
C
C        Get a type 03 PCK record for the interval ET, ET+STEP
C
         CALL GETREC ( ET, ET+STEP, RECORD )
C
C        Add the record to the segment.
C
         CALL PCK03A ( HANDLE, 1, RECORD, ET )
C
C        Increment the time interval.
C
         ET = ET + STEP

      END DO
C
C     End the segment.
```

```
C
      CALL PCK03E ( HANDLE )
C
C     Close the PCK file.
C
      CALL PCKCLS ( HANDLE )
```

Summary of Calling Sequences

Loading files:

```
LDPOOL ( FILE )          for text PCK kernels
PCKLOF ( FILE, HANDLE )  for binary PCK kernels
```

Unloading binary PCK files:

```
PCKUOF ( HANDLE )       for binary PCK kernels
```

Searching binary PCK files for appropriate information:

```
PCKSFS ( BODY, TIME, HANDLE, DESCR, IDENT, FOUND )
```

Testing for the presence of variables in the text kernel pool:

```
BODFND ( BODY, ITEM )
```

Obtaining values associated with variables from the text kernel pool:

```
BODVAR ( NAME, BODY, N,      VALUES )
RTPOOL ( NAME, N,    VALUES, FOUND  )
```

Obtaining Euler angles and their derivatives from a binary PCK file:

```
PCKEUL ( BODY, ET, FOUND, REF, EULANG )
```

Obtaining inertial-to-body-fixed coordinate transformation matrices:

```
TIPBOD ( REF, BODY, ET, TIPM )
BODMAT (      BODY, ET, TIPM )
```

Obtaining Euler angles defining inertial-to-body-fixed coordinate transformations:

Obtaining inertial-to-body-fixed state transformation matrices:

```
TISBOD ( REF, BODY, ET, TSIPM )
```

Opening a binary PCK file for writing:

```
PCKOPN ( NAME, IFNAME, NCOMCH, HANDLE )
```

Writing type 2 segment to a binary PCK file:

```
PCKW02 ( HANDLE, BODY,   FRAME,  FIRST, LAST, SEGID, INTLEN,
         N,      POLYDG, CDATA,  BTIME )
```

Writing type 3 segment to a binary PCK file:

```
To begin a type 3 segment:
PCK03B ( HANDLE, SEGID, BODY, FRAME, FIRST,  LAST, CHBDEG )

To add data to a type 3 sgement:
PCK03A ( HANDLE, NRECS, RECRDS, EPOCHS )

To end a type 3 segment:
PCK03E ( HANDLE )
```

Close a binary PCK file after writing:

```
PCKCLS ( HANDLE )
```

Reading a type 2 segment from a binary PCK file:

```
PCKE02 ( ET, RECORD, EULANG )
```

Evaluating a type 2 segment from a binary PCK file:

```
PCKE02 ( ET, RECORD, EULANG )
```

Reading a type 3 segment from a binary PCK file:

```
PCKR03 ( HANDLE, DESCR, ET, RECORD )
```

Evaluating a type 3 segment from a binary PCK file:

```
PCKE03 ( ET, RECORD, ROTMAT )
```

Appendix A: Sample Text PCK file

The file shown here is an example of a text PCK file. This file is intended only for use as an example; it should not be used as a source of data.

The label set shown here is (as the comments in the PCK file indicate) a draft set. The label set is shown to complete the qualitative description of the PCK format; PCKs produced by NAIF will contain label sets, but the form and contents of those sets will differ from that of the label set shown below.

```
\beginlabel

; ***********DRAFT**DRAFT**DRAFT**DRAFT**DRAFT*****************
; This is a draft and incomplete prototype label set, for
; discussion.
; Do NOT write any code using this information or assuming the
; structure used here.

; PDS Labels
PDS_VERSION_ID                       = PDS3
RECORD_TYPE                          = STREAM
RECORD_BYTES                         = n/a
^SPICE_KERNEL                        = "value"
MISSION_NAME                         = GALILEO
SPACECRAFT_NAME                      = GALILEO_ORBITER
DATA_SET_ID                          = "value"
KERNEL_TYPE_ID                       = PCK
PRODUCT_ID                           = "value"
ORIGINAL_FILE_NAME                   = "value"
PRODUCT_CREATION_TIME                = 1994-03-30T16:45:00
PRODUCER_ID                          = PDS_NAIF
MISSION_PHASE_NAME                   = value
MISSION_PHASE_TYPE                   = value
PRODUCT_VERSION_TYPE                 = value
PRODUCT_VERSION_ID                   = value
HARDWARE_MODEL_ID                    = "value"
SOFTWARE_VERSION_ID                  = "value"
PLATFORM_OR_MOUNTING_NAME            = n/a
FLIGHT_SEQUENCE_NAME                 = "value"
START_TIME                           = n/a
STOP_TIME                            = n/a
SPACECRAFT_CLOCK_START_COUNT         = n/a
SPACECRAFT_CLOCK_STOP_COUNT          = n/a
TARGET_NAME                          = n/a
INSTRUMENT_NAME                      = n/a
INSTRUMENT_ID                        = n/a
```

```
SOURCE_PRODUCT_ID                     = "value"
NOTE                                  = "value"
OBJECT                                = SPICE_KERNEL
INTERCHANGE_FORMAT                    = ASCII
KERNEL_TYPE                           = TARGET_CONSTANTS
DESCRIPTION                           = 1991 IAU cartgraphic values
END_OBJECT                            = SPICE_KERNEL
END

; end PDS Labels

\endlabel



P_constants (PcK) SPICE kernel file
===================================================================

Update by: Karen Zukor (NAIF)    1994 March 30

     First draft of a PCK with internal labels included.
     NOTE: some of the label keywords, values, syntax and overall
     organization are likely to change. Do not write any code that
     uses the above labels.


Organization
--------------------------------------------------------

     The sections of this file are as follows.

     Introductory Information:

         --   Version description

         --   Disclaimer

         --   Sources

         --   Explanation of the PcK contents

         --   Body numbers and names

     Pck Data:

         --   Orientation constants for the Sun and planets

         --   Orientation constants for satellites

         --   Radii for all bodies


Version description
--------------------------------------------------------

     This is a prototype PcK, for use in SPICE training and
     constants evaluation only.

     This file is based on the 1991 IAU report,
```

```
     NAIF document [306].


Disclaimer
--------------------------------------------------------

     This constants file may not contain the parameter values that
     you prefer. Note that this file may be readily modified by
     you or anyone else. NAIF suggests that you inspect this file
     visually before proceeding with any critical or extended data
     processing.

     NAIF requests that you update the `by line' and date if
     you modify the file.


Sources
--------------------------------------------------------

     The sources for the constants listed in this file are:

         1.   ``Report of the IAU/IAG/COSPAR Working Group on
              Cartographic Coordinates and Rotational Elements of
              the Planets and Satellites: 1991,'' March 3, 1992.

         2.   ``The Astronomical Almanac,'' 1990.

         3.   ``Planetary Geodetic Control Using Satellite
              Imaging,'' Journal of Geophysical Research, Vol. 84,
              No. B3, March 10, 1979, by Thomas C. Duxbury. This
              paper is cataloged as NAIF document 190.0.

         4.   Letter from Thomas C. Duxbury to Dr. Ephraim
              Lazeryevich Akim, Keldish Institute of Applied
              Mathematics, USSR Academy of Sciences, Moscow,
              USSR. This letter is catalogued as NAIF document
              number 195.0.

         5.   Mars Observer Planetary Constants and Models,
              November 1990, Mars Observer Document No. 642-321,
              JPL Document No. D-3444.

     Most values are from [1], and these are generally taken from
     [1].  All exceptions are commented where they occur in this
     file. The exceptions are:

         --   The second nutation precession angle (M2) for Mars is
              represented by a quadratic polynomial in the 1991
              IAU report.   The SPICE subroutine BODEUL can not
              handle this term (which is extremely small), so we
              truncate the polynomial to a linear one.

         --   The expression for the prime meridian of Deimos (
              body 402 ) includes a cosine term, which BODEUL
              doesn't yet handle. The amplitude of the term is
              0.19 degrees. We simply ignore this term.
```

```
Body numbers and names
(Contains only information pertinent to Mars Observer)
----------------------------------------------------------

   Barycenters:

        3  Earth barycenter
        4  Mars barycenter

        While not relevant to the P_constants kernel, we note here
        for completeness that 0 is used to represent the solar
        system barycenter.


   Mass centers:

        399 Earth

        401 Phobos
        402 Deimos
        499 Mars

        10  Sun


Orientation constants for the Sun and planets
--------------------------------------------------------


Sun

        \begindata

        BODY10_POLE_RA         = (  286.13       0.          0. )
        BODY10_POLE_DEC        = (   63.87       0.          0. )
        BODY10_PM              = (   84.10     +14.18440     0. )
        BODY10_LONG_AXIS       = (    0.                        )


        \begintext

Earth

        \begindata

        BODY399_POLE_RA        = (    0.      -0.641         0. )
        BODY399_POLE_DEC       = (  +90.      -0.557         0. )
        BODY399_PM             = (  190.16  +360.9856235     0. )
        BODY399_LONG_AXIS      = (    0.                        )

        \begintext

        The linear terms before and after scaling:

           -.052992   -->     -1935.5328
           -.105984   -->     -3871.0656
         -13.012      -->   -475263.3
          13.340716   -->   +487269.6519
           -.9856     -->    -35999.04
```

```
   \begindata

   BODY3_NUT_PREC_ANGLES  = (  125.045    -1935.5328
                               250.090    -3871.0656
                               260.008   475263.3
                               176.625   487269.6519
                               357.529    35999.04     )


   \begintext

Mars


   \begindata

   BODY499_POLE_RA          = (  317.681     -0.108       0. )
   BODY499_POLE_DEC         = (  +52.886     -0.061       0. )
   BODY499_PM               = (  176.868   +350.8919830   0. )

   \begintext

Source [3] specifies the following value for the lambda_a
term ( BODY4_LONG_AXIS ) for Mars.

This term is the POSITIVE WEST LONGITUDE, measured from the
prime meridian, of the longest axis of the ellipsoid
representing the 'mean planet surface', as the article states.

      body499_long_axis        = (  110.  )

Source [4] specifies the lambda_a value

      body499_long_axis        = (  104.9194  )

We list these lambda_a values for completeness. [5] gives
equal values for both equatorial radii, so the lambda_a
offset does not apply.

The 1991 IAU report defines M2, the second nutation
precession angle, by:

                                               2
   192.93  +  1128.4096700 d  +  8.864 T

We truncate the M2 series to a linear expression.

Again, the linear terms are scaled by 36525.0:

    -0.4357640000000000      -->     -15916.28010000000
      1128.409670000000      -->   41215163.19675000
    -1.8151000000000000E-02  -->       -662.9652750000000

   \begindata
   BODY4_NUT_PREC_ANGLES  = (  169.51     -15916.2801
                               192.93  +41215163.19675
                                53.47       -662.965275  )
```

```
        \begintext


Orientation constants for the satellites
--------------------------------------------------------


Satellites of Mars

     Phobos:

     The quadratic prime meridian term is scaled by 1/36525**2:

        8.864000000000000   --->   6.6443009930565219E-09

        \begindata


        BODY401_POLE_RA      = (317.68    -0.108     0.           )
        BODY401_POLE_DEC     = (+52.90    -0.061     0.           )
        BODY401_PM           = ( 35.06 +1128.8445850 8.864        )
        BODY401_LONG_AXIS    = (  0.                              )

        BODY401_NUT_PREC_RA  = ( +1.79   0.    0.  )
        BODY401_NUT_PREC_DEC = ( -1.08   0.    0.  )
        BODY401_NUT_PREC_PM  = ( -1.42  -0.78  0.  )


        \begintext

     Deimos:

     There's a new wrinkle in the Deimos prime meridian
     expression, which is:
                                                   2
           W = 79.41  +  285.1618970 d  -  0.520 T  -  2.58 sin M
                                                                3

                                                    +  0.19 cos M .
                                                                3

                                                              ^
                                                     (new wrinkle)


     At the present time, the constants kernel software (the
     routine bodeul_ in particular) cannot handle the cosine term;
     we omit this term for the time being. The maximum error we
     can make is 0.19 degrees.

     The quadratic prime meridian term is scaled by 1/36525**2:

         -0.5200000000000000  --->   -3.8978300049519307E-10

        \begindata

        BODY402_POLE_RA      = (316.65   -0.108      0.           )
```

```
        BODY402_POLE_DEC     = (+53.52   -0.061      0.           )
        BODY402_PM           = ( 79.41 +285.1618970 -3.897830D-10 )
        BODY402_LONG_AXIS    = (  0.                              )

        BODY402_NUT_PREC_RA  = (  0.   0.   +2.98  )
        BODY402_NUT_PREC_DEC = (  0.   0.   -1.78  )
        BODY402_NUT_PREC_PM  = (  0.   0.   -2.58  )


        \begintext


Radii of bodies
--------------------------------------------------------


Sun

     Value for the Sun is from the 1990 Almanac (page K7).

        \begindata

        BODY10_RADII      = (   696000.   696000.   696000.   )

        \begintext


Earth

     Values for the Earth are unchanged in the 1991 IAU report.

        \begindata

        BODY399_RADII     = (   6378.140   6378.140    6356.75   )

        \begintext


Mars


     Current values taken from [5]:

        \begindata

        BODY499_RADII       = (   3393.4     3393.4     3375.7   )

        \begintext



Satellites of Mars


        \begindata

        BODY401_RADII     = (       13.4        11.2        9.2   )
        BODY402_RADII     = (        7.5         6.1        5.2   )
```