T H E

```
        SSSSS          OOOOOO       FFFFFFFFFFFFF      AAAAAAA
     SSSSSSSSSS     OOOOOOOOOOOO     FFFFFFFFFFFF      AAAAAAAA
    SSSSSSSSSSS   OOOOOOOOOOOOOO    FFFFFFFFFFFF      AAAA AAAA
   SSSS      S   OOOOOO    OOOOO    FFFF             AAAA  AAAA
  SSSSS         OOOOO       OOOO   FFFFF            AAAA   AAAA
  SSSSSSSSSS     OOOO      OOOOO   FFFFFFFFFFFF    AAAA    AAAA
    SSSSSSSSS  OOOOO       OOOO   FFFFFFFFFFFF    AAAAAAAAAAAAA
        SSSSS  OOOO       OOOO    FFFF           AAAAAAAAAAAAAA
 S      SSSS   OOOOO    OOOOO    FFFF           AAAAAAAAAAAAAAA
SSSSSSSSSSS    OOOOOOOOOOOOO     FFFF          AAAA       AAAAA
SSSSSSSSS       OOOOOOOOOO      FFFF          AAAA        AAAAA
  SSSS            OOOOO         FFFF         AAAA         AAAAA
```

S O F T W A R E

L I B R A R I E S

International Astronomical Union

Division A: Fundamental Astronomy

Standards of Fundamental Astronomy Board

Release 19

2023 October 11

CONTENTS

THE IAU-SOFA SOFTWARE LIBRARIES

SOFA stands for "Standards of Fundamental Astronomy". The SOFA software libraries are a collection of subprograms, in source-code form, which implement official IAU algorithms for fundamental-astronomy computations. The subprograms at present comprise 192 "astronomy" routines supported by 55 "vector/matrix" routines, available in both Fortran77 and C implementations.

THE SOFA INITIATIVE

SOFA is an IAU Service which operates as a Standing Working Group under Division A (Fundamental Astronomy).

The IAU set up the SOFA initiative at the 1994 General Assembly, to promulgate an authoritative set of fundamental-astronomy constants and algorithms. At the subsequent General Assembly, in 1997, the appointment of a review board and the selection of a site for the SOFA Center (the outlet for SOFA products) were announced.

The SOFA initiative was originally proposed by the IAU Working Group on Astronomical Standards (WGAS), under the chairmanship of Toshio Fukushima. The proposal was for "...new arrangements to establish and maintain an accessible and authoritative set of constants, algorithms and procedures that implement standard models used in fundamental astronomy". The SOFA Software Libraries implement the "algorithms" part of the SOFA initiative. They were developed under the supervision of an international panel called the SOFA Board. The current membership of this panel is listed in an appendix.

A feature of the original SOFA software proposals was that the products would be self-contained and not depend on other software. This includes basic documentation, which, like the present file, will mostly be plain ASCII text. It should also be noted that there is no assumption that the software will be used on a particular computer and Operating System. Although OS-related facilities may be present (Unix make files for instance, use by the SOFA Center of automatic code management systems, HTML versions of some documentation), the routines themselves will be visible as individual text files and will run on a variety of platforms.

ALGORITHMS

The SOFA Board's initial goal has been to create a set of callable subprograms. Whether "subroutines" or "functions", they are all referred to simply as "routines". They are designed for use by software developers wishing to write complete applications; no runnable, free-standing applications are included in SOFA's present plans.

The algorithms are drawn from a variety of sources. Because most of the routines so far developed have either been standard "text-book" operations or implement well-documented standard algorithms, it has not been necessary to invite the whole community to submit algorithms, though consultation with authorities has occurred where necessary. It should also be noted that consistency with the conventions published by the International Earth Rotation Service was a stipulation in the original SOFA proposals, further constraining the software designs. This state of affairs will continue to exist for some time, as there is a large backlog of agreed extensions to work on. However, in the future the Board may decide to call for proposals, and is in the meantime willing to look into any suggestions that are received by the SOFA Center.

SCOPE

The routines currently available are listed in the next two chapters of this document.

The "astronomy" library comprises 192 routines (plus one obsolete Fortran routine that now appears under a revised name). The areas addressed include calendars, astrometry, time scales, Earth rotation, ephemerides, precession-nutation, star catalog transformations, gnomonic projection, horizon/equatorial transformations and geodetic/geocentric transformations.

The "vector-matrix" library, comprising 55 routines, contains a collection of simple tools for manipulating the vectors, matrices and angles used by the astronomy routines.

There is no explicit commitment by SOFA to support historical models, though as time goes on a legacy of superseded models will naturally accumulate. There is, for example, no support of pre-1976 precession models, though these capabilities could be added were there significant demand.

Though the SOFA software libraries are rather limited in scope, and are likely to remain so for a considerable time, they do offer distinct advantages to prospective users. In particular, the routines are:

* authoritative: they are IAU-backed and have been constructed with great care;

* practical: they are straightforward to use in spite of being precise and rigorous (to some stated degree);

* accessible and supported: they are downloadable from an easy-to-find place, they are in an integrated and consistent form, they come with adequate internal documentation, and help for users is available.

VERSIONS

Once it has been published, an issue is never revised or updated, and remains accessible indefinitely. Subsequent issues may, however, include corrected versions under the original routine name and filenames. However, where a different model is introduced, it will have a different name.

The issues will be referred to by the date when they were announced. The frequency of re-issue will be decided by the Board, taking into account the importance of the changes and the impact on the user community.

DOCUMENTATION

At present there is little free-standing documentation about individual routines. However, each routine has preamble comments which specify in detail what the routine does and how it is used.

The files sofa_pn_f.pdf and sofa_pn_c.pdf (for Fortran and C users respectively) describe the SOFA tools for precession-nutation and other aspects of Earth attitude, and include example code and, in an appendix, diagrams showing the interrelationships between the routines supporting the latest (IAU 2006/2000A) models. Two other pairs of documents introduce time scale transformations (sofa_ts_f.pdf and sofa_ts_c.pdf) and astrometric transformations (sofa_ast_f.pdf and sofa_ast_c.pdf). Finally the two files sofa_vm_f.pdf and sofa_vm_c.pdf describe the vector/matrix routines used throughout SOFA.

PROGRAMMING LANGUAGES AND STANDARDS

The SOFA routines are available in two programming languages at present: Fortran77 and ANSI C. Related software in other languages is under consideration.

The Fortran code conforms to ANSI X3.9-1978 in all but two minor respects: each has an IMPLICIT NONE declaration, and its name has a prefix of "iau_" and may be longer than 6 characters. A global edit to erase both of these will produce ANSI-compliant code with no change in its function.

Coding style, and restrictions on the range of language features, have been much debated by the Board, and the results comply with the majority view. There is (at present) no document that defines the standards, but the code itself offers a wide range of examples of what is acceptable.

The Fortran routines contain explicit numerical constants (the INCLUDE statement is not part of ANSI Fortran77). These are drawn from the file consts.lis, which is listed in an appendix. Constants for the SOFA/C functions are defined in a header file sofam.h.

The naming convention is such that a SOFA routine referred to generically as "EXAMPL" exists as a Fortran subprogram iau_EXAMPL and a C function iauExampl. The calls for the two versions are very similar, with the same arguments in the same order. In a few cases, the C equivalent of a Fortran SUBROUTINE subprogram uses a return value rather than an argument.

Each language version includes a "testbed" main-program that can be used to verify that the SOFA routines have been correctly compiled on the end user's system. The Fortran and C versions are called t_sofa_f.for and t_sofa_c.c respectively. The testbeds execute every SOFA routine and check that the results are within expected accuracy margins. It is not possible to guarantee that all platforms will meet the rather stringent criteria that have been used, and an occasional warning message may be encountered on some systems.

COPYRIGHT ISSUES

Copyright for all of the SOFA software and documentation is owned by the IAU SOFA Board. The Software is made available free of charge for all classes of user, including commercial. However, there are strict rules designed to avoid unauthorized variants coming into circulation. It is permissible to distribute derived works and other modifications, but they must be clearly marked to avoid confusion with the SOFA originals.

Further details are included in the block of comments which concludes every routine. The text is also set out in an appendix to the present document.

ACCURACY

The SOFA policy is to organize the calculations so that the machine accuracy is fully exploited. The gap between the precision of the underlying model or theory and the computational resolution has to be kept as large as possible, hopefully leaving several orders of magnitude of headroom.

The SOFA routines in some cases involve design compromises between rigor and ease of use (and also speed, though nowadays this is seldom a major concern).

ACKNOWLEDGEMENTS

The Board is indebted to a number of contributors, who are acknowledged in the preamble comments of the routines concerned.

The Board's effort is provided by the members' individual institutes.

Resources for operating the SOFA Center are provided by Her Majesty's Nautical Almanac Office, operated by the United Kingdom Hydrographic Office.

SOFA Astronomy Library

PREFACE

The routines described here comprise the SOFA astronomy library. Their general appearance and coding style conforms to conventions agreed by the SOFA Board, and their functions, names and algorithms have been ratified by the Board. Procedures for soliciting and agreeing additions to the library are still evolving.

PROGRAMMING LANGUAGES

The SOFA routines are available in two programming languages at present: Fortran 77 and ANSI C.

Except for a single obsolete Fortran routine, which has no C equivalent, there is a one-to-one relationship between the two language versions. The naming convention is such that a SOFA routine referred to generically as "EXAMPL" exists as a Fortran subprogram iau_EXAMPL and a C function iauExampl. The calls for the two versions are very similar, with the same arguments in the same order. In a few cases, the C equivalent of a Fortran SUBROUTINE subprogram uses a return value rather than an argument.

GENERAL PRINCIPLES

The principal function of the SOFA Astronomy Library is to provide definitive algorithms. A secondary function is to provide software suitable for convenient direct use by writers of astronomical applications.

The astronomy routines call on the SOFA vector/matrix library routines, which are separately listed, and described in sofa_vm_f.pdf (Fortran) and sofa_vm_c.pdf (C).

The routines are designed to exploit the full floating-point accuracy of the machines on which they run, and not to rely on compiler optimizations. Within these constraints, the intention is that the code corresponds to the published formulation (if any).

Dates are always Julian Dates (except in calendar conversion routines) and are expressed as two double precision numbers which sum to the required value.

A distinction is made between routines that implement IAU-approved models and those that use those models to create other results. The former are referred to as "canonical models" in the preamble comments; the latter are described as "support routines".

Using the library requires knowledge of positional astronomy and time-scales. These topics are covered in "Explanatory Supplement to the Astronomical Almanac", 3rd Edition, Sean E. Urban & P. Kenneth Seidelmann (eds.), University Science Books, 2013. Recent developments are documented in the scientific journals, and references to the relevant papers are given in the SOFA code as required. The IERS Conventions are also an essential reference. The routines concerned with Earth attitude (precession-nutation etc.) are described in the SOFA document sofa_pn.pdf. Those concerned with transformations between different time scales are described in sofa_ts_f.pdf (Fortran) and sofa_ts_c.pdf (C). Those concerned with astrometric transformations are described in sofa_ast_f.pdf (Fortran) and sofa_ast_c (C).

ROUTINES

Calendars

```
   CAL2JD    Gregorian calendar to Julian Day number
   EPB       Julian Date to Besselian Epoch
   EPB2JD    Besselian Epoch to Julian Date
   EPJ       Julian Date to Julian Epoch
   EPJ2JD    Julian Epoch to Julian Date
   JD2CAL    Julian Date to Gregorian year, month, day, fraction
   JDCALF    Julian Date to Gregorian date for formatted output

Astrometry

   AB        apply stellar aberration
   APCG      prepare for ICRS <-> GCRS, geocentric, special
   APCG13    prepare for ICRS <-> GCRS, geocentric
   APCI      prepare for ICRS <-> CIRS, terrestrial, special
   APCI13    prepare for ICRS <-> CIRS, terrestrial
   APCO      prepare for ICRS <-> observed, terrestrial, special
   APCO13    prepare for ICRS <-> observed, terrestrial
   APCS      prepare for ICRS <-> CIRS, space, special
   APCS13    prepare for ICRS <-> CIRS, space
   APER      insert ERA into context
   APER13    update context for Earth rotation
   APIO      prepare for CIRS <-> observed, terrestrial, special
   APIO13    prepare for CIRS <-> observed, terrestrial
   ATCC13    catalog -> astrometric
   ATCCQ     quick catalog -> astrometric
   ATCI13    catalog -> CIRS
   ATCIQ     quick ICRS -> CIRS
   ATCIQN    quick ICRS -> CIRS, multiple deflections
   ATCIQZ    quick astrometric ICRS -> CIRS
   ATCO13    ICRS -> observed
   ATIC13    CIRS -> ICRS
   ATICQ     quick CIRS -> ICRS
   ATICQN    quick CIRS -> ICRS, multiple deflections
   ATIO13    CIRS -> observed
   ATIOQ     quick CIRS -> observed
   ATOC13    observed -> astrometric ICRS
   ATOI13    observed -> CIRS
   ATOIQ     quick observed -> CIRS
   LD        light deflection by a single solar-system body
   LDN       light deflection by multiple solar-system bodies
   LDSUN     light deflection by the Sun
   PMPX      apply proper motion and parallax
   PMSAFE    apply proper motion, with zero-parallax precautions
   PVTOB     observatory position and velocity
   PVSTAR    space motion pv-vector to star catalog data
   REFCO     refraction constants
   STARPM    apply proper motion
   STARPV    star catalog data to space motion pv-vector

Time scales

   D2DTF     format 2-part JD for output
   DAT       Delta(AT) (=TAI-UTC) for a given UTC date
   DTDB      TDB-TT
   DTF2D     encode time and date fields into 2-part JD
   TAITT     TAI to TT
   TAIUT1    TAI to UT1
   TAIUTC    TAI to UTC
   TCBTDB    TCB to TDB
   TCGTT     TCG to TT
   TDBTCB    TDB to TCB
   TDBTT     TDB to TT
   TTTAI     TT to TAI
   TTTCG     TT to TCG
   TTTDB     TT to TDB
   TTUT1     TT to UT1
   UT1TAI    UT1 to TAI
   UT1TT     UT1 to TT
   UT1UTC    UT1 to UTC
   UTCTAI    UTC to TAI
   UTCUT1    UTC to UT1
```

Earth rotation angle and sidereal time

```
   EE00      equation of the equinoxes, IAU 2000
   EE00A     equation of the equinoxes, IAU 2000A
   EE00B     equation of the equinoxes, IAU 2000B
   EE06A     equation of the equinoxes, IAU 2006/2000A
   EECT00    equation of the equinoxes complementary terms, IAU 2000
   EQEQ94    equation of the equinoxes, IAU 1994
   ERA00     Earth rotation angle, IAU 2000
   GMST00    Greenwich mean sidereal time, IAU 2000
   GMST06    Greenwich mean sidereal time, IAU 2006
   GMST82    Greenwich mean sidereal time, IAU 1982
   GST00A    Greenwich apparent sidereal time, IAU 2000A
   GST00B    Greenwich apparent sidereal time, IAU 2000B
   GST06     Greenwich apparent ST, IAU 2006, given NPB matrix
   GST06A    Greenwich apparent sidereal time, IAU 2006/2000A
   GST94     Greenwich apparent sidereal time, IAU 1994
```

Ephemerides (limited precision)

```
   EPV00     Earth position and velocity
   MOON98    Moon position and velocity
   PLAN94    major-planet position and velocity
```

Precession, nutation, polar motion

```
   BI00      frame bias components, IAU 2000
   BP00      frame bias and precession matrices, IAU 2000
   BP06      frame bias and precession matrices, IAU 2006
   BPN2XY    extract CIP X,Y coordinates from NPB matrix
   C2I00A    celestial-to-intermediate matrix, IAU 2000A
   C2I00B    celestial-to-intermediate matrix, IAU 2000B
   C2I06A    celestial-to-intermediate matrix, IAU 2006/2000A
   C2IBPN    celestial-to-intermediate matrix, given NPB matrix, IAU 2000
   C2IXY     celestial-to-intermediate matrix, given X,Y, IAU 2000
   C2IXYS    celestial-to-intermediate matrix, given X,Y and s
   C2T00A    celestial-to-terrestrial matrix, IAU 2000A
   C2T00B    celestial-to-terrestrial matrix, IAU 2000B
   C2T06A    celestial-to-terrestrial matrix, IAU 2006/2000A
   C2TCIO    form CIO-based celestial-to-terrestrial matrix
   C2TEQX    form equinox-based celestial-to-terrestrial matrix
   C2TPE     celestial-to-terrestrial matrix given nutation, IAU 2000
   C2TXY     celestial-to-terrestrial matrix given CIP, IAU 2000
   EO06A     equation of the origins, IAU 2006/2000A
   EORS      equation of the origins, given NPB matrix and s
   FW2M      Fukushima-Williams angles to r-matrix
   FW2XY     Fukushima-Williams angles to X,Y
   LTP       long-term precession matrix
   LTPB      long-term precession matrix, including ICRS frame bias
   LTPECL    long-term precession of the ecliptic
   LTPEQU    long-term precession of the equator
   NUM00A    nutation matrix, IAU 2000A
   NUM00B    nutation matrix, IAU 2000B
   NUM06A    nutation matrix, IAU 2006/2000A
   NUMAT     form nutation matrix
   NUT00A    nutation, IAU 2000A
   NUT00B    nutation, IAU 2000B
   NUT06A    nutation, IAU 2006/2000A
   NUT80     nutation, IAU 1980
   NUTM80    nutation matrix, IAU 1980
   OBL06     mean obliquity, IAU 2006
   OBL80     mean obliquity, IAU 1980
   PB06      zeta,z,theta precession angles, IAU 2006, including bias
   PFW06     bias-precession Fukushima-Williams angles, IAU 2006
   PMAT00    precession matrix (including frame bias), IAU 2000
   PMAT06    PB matrix, IAU 2006
   PMAT76    precession matrix, IAU 1976
   PN00      bias/precession/nutation results, IAU 2000
   PN00A     bias/precession/nutation, IAU 2000A
   PN00B     bias/precession/nutation, IAU 2000B
   PN06      bias/precession/nutation results, IAU 2006
   PN06A     bias/precession/nutation results, IAU 2006/2000A
   PNM00A    classical NPB matrix, IAU 2000A
```

```
   PNM00B    classical NPB matrix, IAU 2000B
   PNM06A    classical NPB matrix, IAU 2006/2000A
   PNM80     precession/nutation matrix, IAU 1976/1980
   P06E      precession angles, IAU 2006, equinox based
   POM00     polar motion matrix
   PR00      IAU 2000 precession adjustments
   PREC76    accumulated precession angles, IAU 1976
   S00       the CIO locator s, given X,Y, IAU 2000A
   S00A      the CIO locator s, IAU 2000A
   S00B      the CIO locator s, IAU 2000B
   S06       the CIO locator s, given X,Y, IAU 2006
   S06A      the CIO locator s, IAU 2006/2000A
   SP00      the TIO locator s', IERS 2003
   XY06      CIP, IAU 2006/2000A, from series
   XYS00A    CIP and s, IAU 2000A
   XYS00B    CIP and s, IAU 2000B
   XYS06A    CIP and s, IAU 2006/2000A

Fundamental arguments for nutation etc.

   FAD03     mean elongation of the Moon from the Sun
   FAE03     mean longitude of Earth
   FAF03     mean argument of the latitude of the Moon
   FAJU03    mean longitude of Jupiter
   FAL03     mean anomaly of the Moon
   FALP03    mean anomaly of the Sun
   FAMA03    mean longitude of Mars
   FAME03    mean longitude of Mercury
   FANE03    mean longitude of Neptune
   FAOM03    mean longitude of the Moon's ascending node
   FAPA03    general accumulated precession in longitude
   FASA03    mean longitude of Saturn
   FAUR03    mean longitude of Uranus
   FAVE03    mean longitude of Venus

Star catalog conversions

   FK52H     transform FK5 star data into the Hipparcos system
   FK5HIP    FK5 to Hipparcos rotation and spin
   FK5HZ     FK5 to Hipparcos assuming zero Hipparcos proper motion
   H2FK5     transform Hipparcos star data into the FK5 system
   HFK5Z     Hipparcos to FK5 assuming zero Hipparcos proper motion
   FK425     transform FK4 star data into FK5
   FK45Z     FK4 to FK5 assuming zero FK5 proper motion
   FK524     transform FK5 star data into FK4
   FK54Z     FK5 to FK4 assuming zero FK5 proper motion

Ecliptic coordinates

   ECEQ06    ecliptic to ICRS, IAU 2006
   ECM06     rotation matrix, ICRS to ecliptic, IAU 2006
   EQEC06    ICRS to ecliptic, IAU 2006
   LTECEQ    ecliptic to ICRS, long term
   LTECM     rotation matrix, ICRS to ecliptic, long-term
   LTEQEC    ICRS to ecliptic, long term

Galactic coordinates

   G2ICRS    transform IAU 1958 galactic coordinates to ICRS
   ICRS2G    transform ICRS coordinates to IAU 1958 Galactic

Geodetic/geocentric

   EFORM     a,f for a nominated Earth reference ellipsoid
   GC2GD     geocentric to geodetic for a nominated ellipsoid
   GC2GDE    geocentric to geodetic given ellipsoid a,f
   GD2GC     geodetic to geocentric for a nominated ellipsoid
   GD2GCE    geodetic to geocentric given ellipsoid a,f

Gnomonic projection

   TPORS     solve for tangent point, spherical
   TPORV     solve for tangent point, vector
```

```
     TPSTS     deproject tangent plane to celestial, spherical
     TPSTV     deproject tangent plane to celestial, vector
     TPXES     project celestial to tangent plane, spherical
     TPXEV     project celestial to tangent plane, vector

  Horizon/equatorial

     AE2HD     (azimuth, altitude) to (hour angle, declination)
     HD2AE     (hour angle, declination) to (azimuth, altitude)
     HD2PA     parallactic angle

  Obsolete

     C2TCEO    former name of C2TCIO


CALLS: FORTRAN VERSION

   CALL iau_AB     ( PNAT, V, S, BM1, PPR )
   CALL iau_AE2HD  ( AZ, EL, PHI, HA, DEC )
   CALL iau_APCG   ( DATE1, DATE2, EB, EH, ASTROM )
   CALL iau_APCG13 ( DATE1, DATE2, ASTROM )
   CALL iau_APCI   ( DATE1, DATE2, EB, EH, X, Y, S, ASTROM )
   CALL iau_APCI13 ( DATE1, DATE2, ASTROM, EO )
   CALL iau_APCO   ( DATE1, DATE2, EB, EH, X, Y, S,
  :                  THETA, ELONG, PHI, HM, XP, YP, SP,
  :                  REFA, REFB, ASTROM )
   CALL iau_APCO13 ( UTC1, UTC2, DUT1, ELONG, PHI, HM, XP, YP,
  :                  PHPA, TC, RH, WL, ASTROM, EO, J )
   CALL iau_APCS   ( DATE1, DATE2, PV, EB, EH, ASTROM )
   CALL iau_APCS13 ( DATE1, DATE2, PV, ASTROM )
   CALL iau_APER   ( THETA, ASTROM )
   CALL iau_APER13 ( UT11, UT12, ASTROM )
   CALL iau_APIO   ( SP, THETA, ELONG, PHI, HM, XP, YP,
  :                  REFA, REFB, ASTROM )
   CALL iau_APIO13 ( UTC1, UTC2, DUT1, ELONG, PHI, HM, XP, YP,
  :                  PHPA, TC, RH, WL, ASTROM, J )
   CALL iau_ATCC13 ( RC, DC, PR, PD, PX, RV, DATE1, DATE2, RA, DA )
   CALL iau_ATCCQ  ( RC, DC, PR, PD, PX, RV, ASTROM, RA, DA )
   CALL iau_ATCI13 ( RC, DC, PR, PD, PX, RV, DATE1, DATE2, RI, DI, EO )
   CALL iau_ATCIQ  ( RC, DC, PR, PD, PX, RV, ASTROM, RI, DI )
   CALL iau_ATCIQN ( RC, DC, PR, PD, PX, RV, ASTROM, N, B, RI, DI )
   CALL iau_ATCIQZ ( RC, DC, ASTROM, RI, DI )
   CALL iau_ATCO13 ( RC, DC, PR, PD, PX, RV, UTC1, UTC2, DUT1, ELONG,
  :                  PHI, HM, XP, YP, PHPA, TC, RH, WL,
  :                  AOB, ZOB, HOB, DOB, ROB, EO, J )
   CALL iau_ATIC13 ( RI, DI, DATE1, DATE2, RC, DC, EO )
   CALL iau_ATICQ  ( RI, DI, ASTROM, RC, DC )
   CALL iau_ATICQN ( RI, DI, ASTROM, N, B, RC, DC )
   CALL iau_ATIO13 ( RI, DI, UTC1, UTC2, DUT1, ELONG, PHI, HM, XP, YP,
  :                  PHPA, TC, RH, WL, AOB, ZOB, HOB, DOB, ROB, J )
   CALL iau_ATIOQ  ( RI, DI, ASTROM, AOB, ZOB, HOB, DOB, ROB )
   CALL iau_ATOC13 ( TYPE, OB1, OB2, UTC1, UTC2, DUT1,
  :                  ELONG, PHI, HM, XP, YP, PHPA, TC, RH, WL,
  :                  RC, DC, J )
   CALL iau_ATOI13 ( TYPE, OB1, OB2, UTC1, UTC2, DUT1,
  :                  ELONG, PHI, HM, XP, YP, PHPA, TC, RH, WL,
  :                  RI, DI, J  )
   CALL iau_ATOIQ  ( TYPE, OB1, OB2, ASTROM, RI, DI )
   CALL iau_BI00   ( DPSIBI, DEPSBI, DRA )
   CALL iau_BP00   ( DATE1, DATE2, RB, RP, RBP )
   CALL iau_BP06   ( DATE1, DATE2, RB, RP, RBP )
   CALL iau_BPN2XY ( RBPN, X, Y )
   CALL iau_C2I00A ( DATE1, DATE2, RC2I )
   CALL iau_C2I00B ( DATE1, DATE2, RC2I )
   CALL iau_C2I06A ( DATE1, DATE2, RC2I )
   CALL iau_C2IBPN ( DATE1, DATE2, RBPN, RC2I )
   CALL iau_C2IXY  ( DATE1, DATE2, X, Y, RC2I )
   CALL iau_C2IXYS ( X, Y, S, RC2I )
   CALL iau_C2T00A ( TTA, TTB, UTA, UTB, XP, YP, RC2T )
   CALL iau_C2T00B ( TTA, TTB, UTA, UTB, XP, YP, RC2T )
   CALL iau_C2T06A ( TTA, TTB, UTA, UTB, XP, YP, RC2T )
   CALL iau_C2TCEO ( RC2I, ERA, RPOM, RC2T )
```

```
 CALL iau_C2TCIO ( RC2I, ERA, RPOM, RC2T )
 CALL iau_C2TEQX ( RBPN, GST, RPOM, RC2T )
 CALL iau_C2TPE  ( TTA, TTB, UTA, UTB, DPSI, DEPS, XP, YP, RC2T )
 CALL iau_C2TXY  ( TTA, TTB, UTA, UTB, X, Y, XP, YP, RC2T )
 CALL iau_CAL2JD ( IY, IM, ID, DJM0, DJM, J )
 CALL iau_D2DTF  ( SCALE, NDP, D1, D2, IY, IM, ID, IHMSF, J )
 CALL iau_DAT    ( IY, IM, ID, FD, DELTAT, J )
 D =  iau_DTDB   ( DATE1, DATE2, UT, ELONG, U, V )
 CALL iau_DTF2D  ( SCALE, IY, IM, ID, IHR, IMN, SEC, D1, D2, J )
 CALL iau_ECEQ06 ( DATE1, DATE2, DL, DB, DR, DD )
 CALL iau_ECM06  ( DATE1, DATE2, RM );
 D =  iau_EE00   ( DATE1, DATE2, EPSA, DPSI )
 D =  iau_EE00A  ( DATE1, DATE2 )
 D =  iau_EE00B  ( DATE1, DATE2 )
 D =  iau_EE06A  ( DATE1, DATE2 )
 D =  iau_EECT00 ( DATE1, DATE2 )
 CALL iau_EFORM  ( N, A, F, J )
 D =  iau_EO06A  ( DATE1, DATE2 )
 D =  iau_EORS   ( RNPB, S )
 D =  iau_EPB    ( DJ1, DJ2 )
 CALL iau_EPB2JD ( EPB, DJM0, DJM )
 D =  iau_EPJ    ( DJ1, DJ2 )
 CALL iau_EPJ2JD ( EPJ, DJM0, DJM )
 CALL iau_EPV00  ( DJ1, DJ2, PVH, PVB, J )
 CALL iau_EQEC06 ( DATE1, DATE2, DR, DD, DL, DB )
 D =  iau_EQEQ94 ( DATE1, DATE2 )
 D =  iau_ERA00  ( DJ1, DJ2 )
 D =  iau_FAD03  ( T )
 D =  iau_FAE03  ( T )
 D =  iau_FAF03  ( T )
 D =  iau_FAJU03 ( T )
 D =  iau_FAL03  ( T )
 D =  iau_FALP03 ( T )
 D =  iau_FAMA03 ( T )
 D =  iau_FAME03 ( T )
 D =  iau_FANE03 ( T )
 D =  iau_FAOM03 ( T )
 D =  iau_FAPA03 ( T )
 D =  iau_FASA03 ( T )
 D =  iau_FAUR03 ( T )
 D =  iau_FAVE03 ( T )
 CALL iau_FK425 ( R1950, D1950, DR1950, DD1950, P1950, V1950,
:                 R2000, D2000, DR2000, DD2000, P2000, V2000 )
 CALL iau_FK45Z ( R1950, D1950, BEPOCH, R2000, D2000 )
 CALL iau_FK524 ( R2000, D2000, DR2000, DD2000, P2000, V2000,
:                 R1950, D1950, DR1950, DD1950, P1950, V1950 )
 CALL iau_FK52H  ( R5, D5, DR5, DD5, PX5, RV5,
:                  RH, DH, DRH, DDH, PXH, RVH )
 CALL iau_FK54Z ( R2000, D2000, BEPOCH, R1950, D1950, DR1950, DD1950 )
 CALL iau_FK5HIP ( R5H, S5H )
 CALL iau_FK5HZ  ( R5, D5, DATE1, DATE2, RH, DH )
 CALL iau_FW2M   ( GAMB, PHIB, PSI, EPS, R )
 CALL iau_FW2XY  ( GAMB, PHIB, PSI, EPS, X, Y )
 CALL iau_G2ICRS ( DL, DB, DR, DD )
 CALL iau_GC2GD  ( N, XYZ, ELONG, PHI, HEIGHT, J )
 CALL iau_GC2GDE ( A, F, XYZ, ELONG, PHI, HEIGHT, J )
 CALL iau_GD2GC  ( N, ELONG, PHI, HEIGHT, XYZ, J )
 CALL iau_GD2GCE ( A, F, ELONG, PHI, HEIGHT, XYZ, J )
 D =  iau_GMST00 ( UTA, UTB, TTA, TTB )
 D =  iau_GMST06 ( UTA, UTB, TTA, TTB )
 D =  iau_GMST82 ( UTA, UTB )
 D =  iau_GST00A ( UTA, UTB, TTA, TTB )
 D =  iau_GST00B ( UTA, UTB )
 D =  iau_GST06  ( UTA, UTB, TTA, TTB, RNPB )
 D =  iau_GST06A ( UTA, UTB, TTA, TTB )
 D =  iau_GST94  ( UTA, UTB )
 CALL iau_H2FK5  ( RH, DH, DRH, DDH, PXH, RVH,
:                  R5, D5, DR5, DD5, PX5, RV5 )
 CALL iau_HD2AE  ( HA, DEC, PHI, AZ, EL )
 D =  iau_HD2PA  ( HA, DEC, PHI )
 CALL iau_HFK5Z  ( RH, DH, DATE1, DATE2, R5, D5, DR5, DD5 )
 CALL iau_ICRS2G ( DR, DD, DL, DB  )
 CALL iau_JD2CAL ( DJ1, DJ2, IY, IM, ID, FD, J )
```

```
 CALL iau_JDCALF ( NDP, DJ1, DJ2, IYMDF, J )
 CALL iau_LD     ( BM, P, Q, E, EM, DLIM, P1 )
 CALL iau_LDN    ( N, B, OB, SC, SN )
 CALL iau_LDSUN  ( P, E, EM, P1 )
 CALL iau_LTECEQ ( EPJ, DL, DB, DR, DD )
 CALL iau_LTECM  ( EPJ, RM] )
 CALL iau_LTEQEC ( EPJ, DR, DD, DL, DB )
 CALL iau_LTP    ( EPJ, RP )
 CALL iau_LTPB   ( EPJ, RPB )
 CALL iau_LTPECL ( EPJ, VEC )
 CALL iau_LTPEQU ( EPJ, VEQ )
 CALL iau_MOON98 ( DATE1, DATE2, PV )
 CALL iau_NUM00A ( DATE1, DATE2, RMATN )
 CALL iau_NUM00B ( DATE1, DATE2, RMATN )
 CALL iau_NUM06A ( DATE1, DATE2, RMATN )
 CALL iau_NUMAT  ( EPSA, DPSI, DEPS, RMATN )
 CALL iau_NUT00A ( DATE1, DATE2, DPSI, DEPS )
 CALL iau_NUT00B ( DATE1, DATE2, DPSI, DEPS )
 CALL iau_NUT06A ( DATE1, DATE2, DPSI, DEPS )
 CALL iau_NUT80  ( DATE1, DATE2, DPSI, DEPS )
 CALL iau_NUTM80 ( DATE1, DATE2, RMATN )
 D =  iau_OBL06  ( DATE1, DATE2 )
 D =  iau_OBL80  ( DATE1, DATE2 )
 CALL iau_PB06   ( DATE1, DATE2, BZETA, BZ, BTHETA )
 CALL iau_PFW06  ( DATE1, DATE2, GAMB, PHIB, PSIB, EPSA )
 CALL iau_PLAN94 ( DATE1, DATE2, NP, PV, J )
 CALL iau_PMAT00 ( DATE1, DATE2, RBP )
 CALL iau_PMAT06 ( DATE1, DATE2, RBP )
 CALL iau_PMAT76 ( DATE1, DATE2, RMATP )
 CALL iau_PMPX   ( RC, DC, PR, PD, PX, RV, PMT, POB, PCO )
 CALL iau_PMSAFE ( RA1, DEC1, PMR1, PMD1, PX1, RV1,
:                  EP1A, EP1B, EP2A, EP2B,
:                  RA2, DEC2, PMR2, PMD2, PX2, RV2, J )
 CALL iau_PN00   ( DATE1, DATE2, DPSI, DEPS,
:                  EPSA, RB, RP, RBP, RN, RBPN )
 CALL iau_PN00A  ( DATE1, DATE2,
:                  DPSI, DEPS, EPSA, RB, RP, RBP, RN, RBPN )
 CALL iau_PN00B  ( DATE1, DATE2,
:                  DPSI, DEPS, EPSA, RB, RP, RBP, RN, RBPN )
 CALL iau_PN06   ( DATE1, DATE2, DPSI, DEPS,
:                  EPSA, RB, RP, RBP, RN, RBPN )
 CALL iau_PN06A  ( DATE1, DATE2,
:                  DPSI, DEPS, RB, RP, RBP, RN, RBPN )
 CALL iau_PNM00A ( DATE1, DATE2, RBPN )
 CALL iau_PNM00B ( DATE1, DATE2, RBPN )
 CALL iau_PNM06A ( DATE1, DATE2, RNPB )
 CALL iau_PNM80  ( DATE1, DATE2, RMATPN )
 CALL iau_P06E   ( DATE1, DATE2,
:                  EPS0, PSIA, OMA, BPA, BQA, PIA, BPIA,
:                  EPSA, CHIA, ZA, ZETAA, THETAA, PA, GAM, PHI, PSI )
 CALL iau_POM00  ( XP, YP, SP, RPOM )
 CALL iau_PR00   ( DATE1, DATE2, DPSIPR, DEPSPR )
 CALL iau_PREC76 ( DATE01, DATE02, DATE11, DATE12, ZETA, Z, THETA )
 CALL iau_PVSTAR ( PV, RA, DEC, PMR, PMD, PX, RV, J )
 CALL iau_PVTOB  ( ELONG, PHI, HM, XP, YP, SP, THETA, PV )
 CALL iau_REFCO  ( PHPA, TC, RH, WL, REFA, REFB )
 D =  iau_S00    ( DATE1, DATE2, X, Y )
 D =  iau_S00A   ( DATE1, DATE2 )
 D =  iau_S00B   ( DATE1, DATE2 )
 D =  iau_S06    ( DATE1, DATE2, X, Y )
 D =  iau_S06A   ( DATE1, DATE2 )
 D =  iau_SP00   ( DATE1, DATE2 )
 CALL iau_STARPM ( RA1, DEC1, PMR1, PMD1, PX1, RV1,
:                  EP1A, EP1B, EP2A, EP2B,
:                  RA2, DEC2, PMR2, PMD2, PX2, RV2, J )
 CALL iau_STARPV ( RA, DEC, PMR, PMD, PX, RV, PV, J )
 CALL iau_TAITT  ( TAI1, TAI2, TT1, TT2, J )
 CALL iau_TAIUT1 ( TAI1, TAI2, DTA, UT11, UT12, J )
 CALL iau_TAIUTC ( TAI1, TAI2, UTC1, UTC2, J )
 CALL iau_TCBTDB ( TCB1, TCB2, TDB1, TDB2, J )
 CALL iau_TCGTT  ( TCG1, TCG2, TT1, TT2, J )
 CALL iau_TDBTCB ( TDB1, TDB2, TCB1, TCB2, J )
 CALL iau_TDBTT  ( TDB1, TDB2, DTR, TT1, TT2, J )
```

```
   CALL iau_TPORS  ( XI, ETA, A, B, A01, B01, A02, B02, N )
   CALL iau_TPORV  ( XI, ETA, V, V01, V02, N )
   CALL iau_TPSTS  ( XI, ETA, A0, B0, A, B )
   CALL iau_TPSTV  ( XI, ETA, V0, V )
   CALL iau_TPXES  ( A, B, A0, B0, XI, ETA, J )
   CALL iau_TPXEV  ( V, V0, XI, ETA, J )
   CALL iau_TTTAI  ( TT1, TT2, TAI1, TAI2, J )
   CALL iau_TTTCG  ( TT1, TT2, TCG1, TCG2, J )
   CALL iau_TTTDB  ( TT1, TT2, DTR, TDB1, TDB2, J )
   CALL iau_TTUT1  ( TT1, TT2, DT, UT11, UT12, J )
   CALL iau_UT1TAI ( UT11, UT12, TAI1, TAI2, J )
   CALL iau_UT1TT  ( UT11, UT12, DT, TT1, TT2, J )
   CALL iau_UT1UTC ( UT11, UT12, DUT, UTC1, UTC2, J )
   CALL iau_UTCTAI ( UTC1, UTC2, DTA, TAI1, TAI2, J )
   CALL iau_UTCUT1 ( UTC1, UTC2, DUT, UT11, UT12, J )
   CALL iau_XY06   ( DATE1, DATE2, X, Y )
   CALL iau_XYS00A ( DATE1, DATE2, X, Y, S )
   CALL iau_XYS00B ( DATE1, DATE2, X, Y, S )
   CALL iau_XYS06A ( DATE1, DATE2, X, Y, S )
```

CALLS: C VERSION

```
       iauAb     ( pnat, v, s, bm1, ppr );
       iauAe2hd  ( az, el, phi, &ha, &dec );
       iauApcg   ( date1, date2, eb, eh, &astrom );
       iauApcg13 ( date1, date2, &astrom );
       iauApci   ( date1, date2, eb, eh, x, y, s, &astrom );
       iauApci13 ( date1, date2, &astrom, &eo );
       iauApco   ( date1, date2, eb, eh, x, y, s,
                   theta, elong, phi, hm, xp, yp, sp,
                   refa, refb, &astrom );
   i = iauApco13 ( utc1, utc2, dut1, elong, phi, hm, xp, yp,
                   phpa, tc, rh, wl, &astrom, &eo );
       iauApcs   ( date1, date2, pv, eb, eh, &astrom );
       iauApcs13 ( date1, date2, pv, &astrom );
       iauAper   ( theta, &astrom );
       iauAper13 ( ut11, ut12, &astrom );
       iauApio   ( sp, theta, elong, phi, hm, xp, yp, refa, refb,
                   &astrom );
   i = iauApio13 ( utc1, utc2, dut1, elong, phi, hm, xp, yp,
                   phpa, tc, rh, wl, &astrom );
       iauAtcc13 ( rc, dc, pr, pd, px, rv, date1, date2, &ra, &da );
       iauAtccq  ( rc, dc, pr, pd, px, rv, &astrom, &ra, &da );
       iauAtci13 ( rc, dc, pr, pd, px, rv, date1, date2,
                   &ri, &di, &eo );
       iauAtciq  ( rc, dc, pr, pd, px, rv, &astrom, &ri, &di );
       iauAtciqn ( rc, dc, pr, pd, px, rv, astrom, n, b, &ri, &di );
       iauAtciqz ( rc, dc, &astrom, &ri, &di );
   i = iauAtco13 ( rc, dc, pr, pd, px, rv, utc1, utc2, dut1,
                   elong phi, hm, xp, yp, phpa, tc, rh, wl,
                   aob, zob, hob, dob, rob, eo );
       iauAtic13 ( ri, di, date1, date2, &rc, &dc, &eo );
       iauAticq  ( ri, di, &astrom, &rc, &dc );
       iauAticqn ( ri, di, astrom, n, b, &rc, &dc );
   i = iauAtio13 ( ri, di, utc1, utc2, dut1, elong, phi, hm, xp, yp,
                   phpa, tc, rh, wl, aob, zob, hob, dob, rob );
       iauAtioq  ( ri, di, &astrom, &aob, &zob, &hob, &dob, &rob );
   i = iauAtoc13 ( type, ob1, ob2, utc1, utc2, dut1,
                   elong, phi, hm, xp, yp, phpa, tc, rh, wl,
                   &rc, &dc );
   i = iauAtoi13 ( type, ob1, ob2, utc1, utc2, dut1, elong, phi, hm,
                   xp, yp, phpa, tc, rh, wl, &ri, &di  );
       iauAtoiq  ( type, ob1, ob2, &astrom, &ri, &di );
       iauBi00   ( &dpsibi, &depsbi, &dra );
       iauBp00   ( date1, date2, rb, rp, rbp );
       iauBp06   ( date1, date2, rb, rp, rbp );
       iauBpn2xy ( rbpn, &x, &y );
       iauC2i00a ( date1, date2, rc2i );
       iauC2i00b ( date1, date2, rc2i );
       iauC2i06a ( date1, date2, rc2i );
       iauC2ibpn ( date1, date2, rbpn, rc2i );
       iauC2ixy  ( date1, date2, x, y, rc2i );
```

```
    iauC2ixys ( x, y, s, rc2i );
    iauC2t00a ( tta, ttb, uta, utb, xp, yp, rc2t );
    iauC2t00b ( tta, ttb, uta, utb, xp, yp, rc2t );
    iauC2t06a ( tta, ttb, uta, utb, xp, yp, rc2t );
    iauC2tcio ( rc2i, era, rpom, rc2t );
    iauC2teqx ( rbpn, gst, rpom, rc2t );
    iauC2tpe  ( tta, ttb, uta, utb, dpsi, deps, xp, yp, rc2t );
    iauC2txy  ( tta, ttb, uta, utb, x, y, xp, yp, rc2t );
i = iauCal2jd ( iy, im, id, &djm0, &djm );
i = iauD2dtf  ( scale, ndp, d1, d2, &iy, &im, &id, ihmsf );
i = iauDat    ( iy, im, id, fd, &deltat );
d = iauDtdb   ( date1, date2, ut, elong, u, v );
i = iauDtf2d  ( scale, iy, im, id, ihr, imn, sec, &d1, &d2 );
    iauEceq06 ( date1, date2, dl, db, &dr, &dd );
    iauEcm06  ( date1, date2, rm );
d = iauEe00   ( date1, date2, epsa, dpsi );
d = iauEe00a  ( date1, date2 );
d = iauEe00b  ( date1, date2 );
d = iauEe06   ( date1, date2 );
d = iauEect00 ( date1, date2 );
i = iauEform  ( n, &a, &f );
d = iauEo06   ( date1, date2 );
d = iauEors   ( rnpb, s );
d = iauEpb    ( dj1, dj2 );
    iauEpb2jd ( epb, &djm0, &djm );
d = iauEpj    ( dj1, dj2 );
    iauEpj2jd ( epj, &djm0, &djm );
i = iauEpv00  ( dj1, dj2, pvh, pvb );
    iauEqec06 ( date1, date2, dr, dd, &dl, &db );
d = iauEqeq94 ( date1, date2 );
d = iauEra00  ( dj1, dj2 );
d = iauFad03  ( t );
d = iauFae03  ( t );
d = iauFaf03  ( t );
d = iauFaju03 ( t );
d = iauFal03  ( t );
d = iauFalp03 ( t );
d = iauFama03 ( t );
d = iauFame03 ( t );
d = iauFane03 ( t );
d = iauFaom03 ( t );
d = iauFapa03 ( t );
d = iauFasa03 ( t );
d = iauFaur03 ( t );
d = iauFave03 ( t );
    iauFk425 ( r1950, d1950, dr1950, dd1950, p1950, v1950,
               &r2000, &d2000, &dr2000, &dd2000, &p2000, &v2000 );
    iauFk45z ( r1950, d1950, bepoch, &r2000, &d2000 );
    iauFk524 ( r2000, d2000, dr2000, dd2000, p2000, v2000,
               &r1950, &d1950, &dr1950, &dd1950, &p1950, &v1950 );
    iauFk52h  ( r5, d5, dr5, dd5, px5, rv5,
                &rh, &dh, &drh, &ddh, &pxh, &rvh );
    iauFk54z ( r2000, d2000, bepoch,
               &r1950, &d1950, &dr1950, &dd1950 );
    iauFk5hip ( r5h, s5h );
    iauFk5hz  ( r5, d5, date1, date2, &rh, &dh );
    iauFw2m   ( gamb, phib, psi, eps, r );
    iauFw2xy  ( gamb, phib, psi, eps, &x, &y );
    iauG2icrs ( dl, db, &dr, &dd );
i = iauGc2gd  ( n, xyz, &elong, &phi, &height );
i = iauGc2gde ( a, f, xyz, &elong, &phi, &height );
i = iauGd2gc  ( n, elong, phi, height, xyz );
i = iauGd2gce ( a, f, elong, phi, height, xyz );
d = iauGmst00 ( uta, utb, tta, ttb );
d = iauGmst06 ( uta, utb, tta, ttb );
d = iauGmst82 ( uta, utb );
d = iauGst00a ( uta, utb, tta, ttb );
d = iauGst00b ( uta, utb );
d = iauGst06  ( uta, utb, tta, ttb, rnpb );
d = iauGst06a ( uta, utb, tta, ttb );
d = iauGst94  ( uta, utb );
    iauH2fk5  ( rh, dh, drh, ddh, pxh, rvh,
                &r5, &d5, &dr5, &dd5, &px5, &rv5 );
```

```
    iauHd2ae  ( ha, dec, phi, &az, &el );
d = iauHd2pa  ( ha, dec, phi );
    iauHfk5z  ( rh, dh, date1, date2,
                &r5, &d5, &dr5, &dd5 );
    iauIcrs2g ( dr, dd, &dl, &db );
i = iauJd2cal ( dj1, dj2, &iy, &im, &id, &fd );
i = iauJdcalf ( ndp, dj1, dj2, iymdf );
    iauLd     ( bm, p, q, e, em, dlim, p1 );
    iauLdn    ( n, b, ob, sc, sn );
    iauLdsun  ( p, e, em, p1 );
    iauLteceq ( epj,  dl, db, &dr, &dd );
    iauLtecm  ( epj,  rm );
    iauLteqec ( epj,  dr,  dd, &dl, &db );
    iauLtp    ( epj,  rp );
    iauLtpb   ( epj,  rpb );
    iauLtpecl ( epj,  vec );
    iauLtpequ ( epj,  veq );
    iauMoon98 ( date1, date2, pv );
    iauNum00a ( date1, date2, rmatn );
    iauNum00b ( date1, date2, rmatn );
    iauNum06a ( date1, date2, rmatn );
    iauNumat  ( epsa, dpsi, deps, rmatn );
    iauNut00a ( date1, date2, &dpsi, &deps );
    iauNut00b ( date1, date2, &dpsi, &deps );
    iauNut06a ( date1, date2, &dpsi, &deps );
    iauNut80  ( date1, date2, &dpsi, &deps );
    iauNutm80 ( date1, date2, rmatn );
d = iauObl06  ( date1, date2 );
d = iauObl80  ( date1, date2 );
    iauPb06   ( date1, date2, &bzeta, &bz, &btheta );
    iauPfw06  ( date1, date2, &gamb, &phib, &psib, &epsa );
i = iauPlan94 ( date1, date2, np, pv );
    iauPmat00 ( date1, date2, rbp );
    iauPmat06 ( date1, date2, rbp );
    iauPmat76 ( date1, date2, rmatp );
    iauPmpx   ( rc, dc, pr, pd, px, rv, pmt, pob, pco );
i = iauPmsafe ( ra1, dec1, pmr1, pmd1, px1, rv1,
                ep1a, ep1b, ep2a, ep2b,
                &ra2, &dec2, &pmr2, &pmd2, &px2, &rv2);
    iauPn00   ( date1, date2, dpsi, deps,
                &epsa, rb, rp, rbp, rn, rbpn );
    iauPn00a  ( date1, date2,
                &dpsi, &deps, &epsa, rb, rp, rbp, rn, rbpn );
    iauPn00b  ( date1, date2,
                &dpsi, &deps, &epsa, rb, rp, rbp, rn, rbpn );
    iauPn06   ( date1, date2, dpsi, deps,
                &epsa, rb, rp, rbp, rn, rbpn );
    iauPn06a  ( date1, date2,
                &dpsi, &deps, &epsa, rb, rp, rbp, rn, rbpn );
    iauPnm00a ( date1, date2, rbpn );
    iauPnm00b ( date1, date2, rbpn );
    iauPnm06a ( date1, date2, rnpb );
    iauPnm80  ( date1, date2, rmatpn );
    iauP06e   ( date1, date2,
                &eps0, &psia, &oma, &bpa, &bqa, &pia, &bpia,
                &epsa, &chia, &za, &zetaa, &thetaa, &pa,
                &gam, &phi, &psi );
    iauPom00  ( xp, yp, sp, rpom );
    iauPr00   ( date1, date2, &dpsipr, &depspr );
    iauPrec76 ( date01, date02, date11, date12, &zeta, &z, &theta );
i = iauPvstar ( pv, &ra, &dec, &pmr, &pmd, &px, &rv );
    iauPvtob  ( elong, phi, hm, xp, yp, sp, theta, pv );
    iauRefco  ( phpa, tc, rh, wl, refa, refb );
d = iauS00    ( date1, date2, x, y );
d = iauS00a   ( date1, date2 );
d = iauS00b   ( date1, date2 );
d = iauS06    ( date1, date2, x, y );
d = iauS06a   ( date1, date2 );
d = iauSp00   ( date1, date2 );
i = iauStarpm ( ra1, dec1, pmr1, pmd1, px1, rv1,
                ep1a, ep1b, ep2a, ep2b,
                &ra2, &dec2, &pmr2, &pmd2, &px2, &rv2 );
i = iauStarpv ( ra, dec, pmr, pmd, px, rv, pv );
```

```
i = iauTaitt  ( tai1, tai2, &tt1, &tt2 );
i = iauTaiut1 ( tai1, tai2, dta, &ut11, &ut12 );
i = iauTaiutc ( tai1, tai2, &utc1, &utc2 );
i = iauTcbtdb ( tcb1, tcb2, &tdb1, &tdb2 );
i = iauTcgtt  ( tcg1, tcg2, &tt1, &tt2 );
i = iauTdbtcb ( tdb1, tdb2, &tcb1, &tcb2 );
i = iauTdbtt  ( tdb1, tdb2, dtr, &tt1, &tt2 );
i = iauTpors  ( xi, eta, a, b, &a01, &b01, &a02, &b02 );
i = iauTporv  ( xi, eta, v, v01, v02 );
    iauTpsts  ( xi, eta, a0, b0, &a, &b );
    iauTpstv  ( xi, eta, v0, v );
i = iauTpxes  ( a, b, a0, b0, &xi, &eta );
i = iauTpxev  ( v, v0, &xi, &eta );
i = iauTttai  ( tt1, tt2, &tai1, &tai2 );
i = iauTttcg  ( tt1, tt2, &tcg1, &tcg2 );
i = iauTttdb  ( tt1, tt2, dtr, &tdb1, &tdb2 );
i = iauTtut1  ( tt1, tt2, dt, &ut11, &ut12 );
i = iauUt1tai ( ut11, ut12, &tai1, &tai2 );
i = iauUt1tt  ( ut11, ut12, dt, &tt1, &tt2 );
i = iauUt1utc ( ut11, ut12, dut, &utc1, &utc2 );
i = iauUtctai ( utc1, utc2, dta, &tai1, &tai2 );
i = iauUtcut1 ( utc1, utc2, dut, &ut11, &ut12 );
    iauXy06   ( date1, date2, &x, &y );
    iauXys00a ( date1, date2, &x, &y, &s );
    iauXys00b ( date1, date2, &x, &y, &s );
    iauXys06a ( date1, date2, &x, &y, &s );
```

SOFA Vector/Matrix Library

PREFACE

The routines described here comprise the SOFA vector/matrix library. Their general appearance and coding style conforms to conventions agreed by the SOFA Board, and their functions, names and algorithms have been ratified by the Board. Procedures for soliciting and agreeing additions to the library are still evolving.

PROGRAMMING LANGUAGES

The SOFA routines are available in two programming languages at present: Fortran 77 and ANSI C.

There is a one-to-one relationship between the two language versions. The naming convention is such that a SOFA routine referred to generically as "EXAMPL" exists as a Fortran subprogram iau_EXAMPL and a C function iauExampl. The calls for the two versions are very similar, with the same arguments in the same order. In a few cases, the C equivalent of a Fortran SUBROUTINE subprogram uses a return value rather than an argument.

GENERAL PRINCIPLES

The library consists mostly of routines which operate on ordinary Cartesian vectors (x,y,z) and 3x3 rotation matrices. However, there is also support for vectors which represent velocity as well as position and vectors which represent rotation instead of position. The vectors which represent both position and velocity may be considered still to have dimensions (3), but to comprise elements each of which is two numbers, representing the value itself and the time derivative. Thus:

* "Position" or "p" vectors (or just plain 3-vectors) have dimension (3) in Fortran and [3] in C.

* "Position/velocity" or "pv" vectors have dimensions (3,2) in Fortran and [2][3] in C.

* "Rotation" or "r" matrices have dimensions (3,3) in Fortran and [3][3] in C. When used for rotation, they are "orthogonal"; the inverse of such a matrix is equal to the transpose. Most of the routines in this library do not assume that r-matrices are necessarily orthogonal and in fact work on any 3x3 matrix.

* "Rotation" or "r" vectors have dimensions (3) in Fortran and [3] in C. Such vectors are a combination of the Euler axis and angle and are convertible to and from r-matrices. The direction is the axis of rotation and the magnitude is the angle of rotation, in radians. Because the amount of rotation can be scaled up and down simply by multiplying the vector by a scalar, r-vectors are useful for representing spins about an axis which is fixed.

* The above rules mean that in terms of memory address, the three velocity components of a pv-vector follow the three position components. Application code is permitted to exploit this and all other knowledge of the internal layouts: that x, y and z appear in that order and are in a right-handed Cartesian coordinate system etc. For example, the cp function (copy a p-vector) can be used to copy the velocity component of a pv-vector (indeed, this is how the CPV routine is coded).

* The routines provided do not completely fill the range of operations that link all the various vector and matrix options, but are confined to functions that are required by other parts of the SOFA software or which are likely to prove useful.

In addition to the vector/matrix routines, the library contains some routines related to spherical angles, including conversions to and from sexagesimal format.

Using the library requires knowledge of vector/matrix methods, spherical trigonometry, and methods of attitude representation. These topics are covered in many textbooks, including "Spacecraft Attitude Determination and Control", James R. Wertz (ed.), Astrophysics and Space Science Library, Vol. 73, D. Reidel Publishing Company, 1986.

```
OPERATIONS INVOLVING P-VECTORS AND R-MATRICES

  Initialize

     ZP        zero p-vector
     ZR        initialize r-matrix to null
     IR        initialize r-matrix to identity

  Copy

     CP        copy p-vector
     CR        copy r-matrix

  Build rotations

     RX        rotate r-matrix about x
     RY        rotate r-matrix about y
     RZ        rotate r-matrix about z

  Spherical/Cartesian conversions

     S2C       spherical to unit vector
     C2S       unit vector to spherical
     S2P       spherical to p-vector
     P2S       p-vector to spherical

  Operations on vectors

     PPP       p-vector plus p-vector
     PMP       p-vector minus p-vector
     PPSP      p-vector plus scaled p-vector
     PDP       inner (=scalar=dot) product of two p-vectors
     PXP       outer (=vector=cross) product of two p-vectors
     PM        modulus of p-vector
     PN        normalize p-vector returning modulus
     SXP       multiply p-vector by scalar

  Operations on matrices

     RXR       r-matrix multiply
     TR        transpose r-matrix

  Matrix-vector products

     RXP       product of r-matrix and p-vector
     TRXP      product of transpose of r-matrix and p-vector

  Separation and position-angle

     SEPP      angular separation from p-vectors
     SEPS      angular separation from spherical coordinates
     PAP       position-angle from p-vectors
     PAS       position-angle from spherical coordinates

  Rotation vectors

     RV2M      r-vector to r-matrix
     RM2V      r-matrix to r-vector


OPERATIONS INVOLVING PV-VECTORS
```

```
  Initialize

     ZPV       zero pv-vector

  Copy/extend/extract

     CPV       copy pv-vector
     P2PV      append zero velocity to p-vector
     PV2P      discard velocity component of pv-vector

  Spherical/Cartesian conversions

     S2PV      spherical to pv-vector
     PV2S      pv-vector to spherical

  Operations on pv-vectors

     PVPPV     pv-vector plus pv-vector
     PVMPV     pv-vector minus pv-vector
     PVDPV     inner (=scalar=dot) product of two pv-vectors
     PVXPV     outer (=vector=cross) product of two pv-vectors
     PVM       modulus of pv-vector
     SXPV      multiply pv-vector by scalar
     S2XPV     multiply pv-vector by two scalars
     PVU       update pv-vector
     PVUP      update pv-vector discarding velocity

  Matrix-vector products

     RXPV      product of r-matrix and pv-vector
     TRXPV     product of transpose of r-matrix and pv-vector


OPERATIONS ON ANGLES

  Wrap

     ANP       normalize radians to range 0 to 2pi
     ANPM      normalize radians to range -pi to +pi

  To sexagesimal

     A2TF      decompose radians into hours, minutes, seconds
     A2AF      decompose radians into degrees, arcminutes, arcseconds
     D2TF      decompose days into hours, minutes, seconds

  From sexagesimal

     AF2A      degrees, arcminutes, arcseconds to radians
     TF2A      hours, minutes, seconds to radians
     TF2D      hours, minutes, seconds to days


CALLS: FORTRAN VERSION

   CALL iau_A2AF  ( NDP, ANGLE, SIGN, IDMSF )
   CALL iau_A2TF  ( NDP, ANGLE, SIGN, IHMSF )
   CALL iau_AF2A  ( S, IDEG, IAMIN, ASEC, RAD, J )
   D =  iau_ANP   ( A )
   D =  iau_ANPM  ( A )
   CALL iau_C2S   ( P, THETA, PHI )
   CALL iau_CP    ( P, C )
   CALL iau_CPV   ( PV, C )
   CALL iau_CR    ( R, C )
   CALL iau_D2TF  ( NDP, DAYS, SIGN, IHMSF )
   CALL iau_IR    ( R )
   CALL iau_P2PV  ( P, PV )
   CALL iau_P2S   ( P, THETA, PHI, R )
   CALL iau_PAP   ( A, B, THETA )
   CALL iau_PAS   ( AL, AP, BL, BP, THETA )
   CALL iau_PDP   ( A, B, ADB )
   CALL iau_PM    ( P, R )
   CALL iau_PMP   ( A, B, AMB )
```

```
   CALL iau_PN    ( P, R, U )
   CALL iau_PPP   ( A, B, APB )
   CALL iau_PPSP  ( A, S, B, APSB )
   CALL iau_PV2P  ( PV, P )
   CALL iau_PV2S  ( PV, THETA, PHI, R, TD, PD, RD )
   CALL iau_PVDPV ( A, B, ADB )
   CALL iau_PVM   ( PV, R, S )
   CALL iau_PVMPV ( A, B, AMB )
   CALL iau_PVPPV ( A, B, APB )
   CALL iau_PVU   ( DT, PV, UPV )
   CALL iau_PVUP  ( DT, PV, P )
   CALL iau_PVXPV ( A, B, AXB )
   CALL iau_PXP   ( A, B, AXB )
   CALL iau_RM2V  ( R, P )
   CALL iau_RV2M  ( P, R )
   CALL iau_RX    ( PHI, R )
   CALL iau_RXP   ( R, P, RP )
   CALL iau_RXPV  ( R, PV, RPV )
   CALL iau_RXR   ( A, B, ATB )
   CALL iau_RY    ( THETA, R )
   CALL iau_RZ    ( PSI, R )
   CALL iau_S2C   ( THETA, PHI, C )
   CALL iau_S2P   ( THETA, PHI, R, P )
   CALL iau_S2PV  ( THETA, PHI, R, TD, PD, RD, PV )
   CALL iau_S2XPV ( S1, S2, PV )
   CALL iau_SEPP  ( A, B, S )
   CALL iau_SEPS  ( AL, AP, BL, BP, S )
   CALL iau_SXP   ( S, P, SP )
   CALL iau_SXPV  ( S, PV, SPV )
   CALL iau_TF2A  ( S, IHOUR, IMIN, SEC, RAD, J )
   CALL iau_TF2D  ( S, IHOUR, IMIN, SEC, DAYS, J )
   CALL iau_TR    ( R, RT )
   CALL iau_TRXP  ( R, P, TRP )
   CALL iau_TRXPV ( R, PV, TRPV )
   CALL iau_ZP    ( P )
   CALL iau_ZPV   ( PV )
   CALL iau_ZR    ( R )
```

CALLS: C VERSION

```
       iauA2af  ( ndp, angle, &sign, idmsf );
       iauA2tf  ( ndp, angle, &sign, ihmsf );
   i = iauAf2a  ( s, ideg, iamin, asec, &rad );
   d = iauAnp   ( a );
   d = iauAnpm  ( a );
       iauC2s   ( p, &theta, &phi );
       iauCp    ( p, c );
       iauCpv   ( pv, c );
       iauCr    ( r, c );
       iauD2tf  ( ndp, days, &sign, ihmsf );
       iauIr    ( r );
       iauP2pv  ( p, pv );
       iauP2s   ( p, &theta, &phi, &r );
   d = iauPap   ( a, b );
   d = iauPas   ( al, ap, bl, bp );
   d = iauPdp   ( a, b );
   d = iauPm    ( p );
       iauPmp   ( a, b, amb );
       iauPn    ( p, &r, u );
       iauPpp   ( a, b, apb );
       iauPpsp  ( a, s, b, apsb );
       iauPv2p  ( pv, p );
       iauPv2s  ( pv, &theta, &phi, &r, &td, &pd, &rd );
       iauPvdpv ( a, b, adb );
       iauPvm   ( pv, &r, &s );
       iauPvmpv ( a, b, amb );
       iauPvppv ( a, b, apb );
       iauPvu   ( dt, pv, upv );
       iauPvup  ( dt, pv, p );
       iauPvxpv ( a, b, axb );
       iauPxp   ( a, b, axb );
       iauRm2v  ( r, p );
```

```
    iauRv2m  ( p, r );
    iauRx    ( phi, r );
    iauRxp   ( r, p, rp );
    iauRxpv  ( r, pv, rpv );
    iauRxr   ( a, b, atb );
    iauRy    ( theta, r );
    iauRz    ( psi, r );
    iauS2c   ( theta, phi, c );
    iauS2p   ( theta, phi, r, p );
    iauS2pv  ( theta, phi, r, td, pd, rd, pv );
    iauS2xpv ( s1, s2, pv );
d = iauSepp  ( a, b );
d = iauSeps  ( al, ap, bl, bp );
    iauSxp   ( s, p, sp );
    iauSxpv  ( s, pv, spv );
i = iauTf2a  ( s, ihour, imin, sec, &rad );
i = iauTf2d  ( s, ihour, imin, sec, &days );
    iauTr    ( r, rt );
    iauTrxp  ( r, p, trp );
    iauTrxpv ( r, pv, trpv );
    iauZp    ( p );
    iauZpv   ( pv );
    iauZr    ( r );
```

```
void iauA2af(int ndp, double angle, char *sign, int idmsf[4])
/*
**  - - - - - - - -
**   i a u A 2 a f
**  - - - - - - - -
**
**  Decompose radians into degrees, arcminutes, arcseconds, fraction.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     ndp     int     resolution (Note 1)
**     angle   double  angle in radians
**
**  Returned:
**     sign    char*   '+' or '-'
**     idmsf   int[4]  degrees, arcminutes, arcseconds, fraction
**
**  Notes:
**
**  1) The argument ndp is interpreted as follows:
**
**     ndp         resolution
**      :      ...0000 00 00
**     -7         1000 00 00
**     -6          100 00 00
**     -5           10 00 00
**     -4            1 00 00
**     -3            0 10 00
**     -2            0 01 00
**     -1            0 00 10
**      0            0 00 01
**      1            0 00 00.1
**      2            0 00 00.01
**      3            0 00 00.001
**      :            0 00 00.000...
**
**  2) The largest positive useful value for ndp is determined by the
**     size of angle, the format of doubles on the target platform, and
**     the risk of overflowing idmsf[3].  On a typical platform, for
**     angle up to 2pi, the available floating-point precision might
**     correspond to ndp=12.  However, the practical limit is typically
**     ndp=9, set by the capacity of a 32-bit int, or ndp=4 if int is
**     only 16 bits.
**
**  3) The absolute value of angle may exceed 2pi.  In cases where it
**     does not, it is up to the caller to test for and handle the
**     case where angle is very nearly 2pi and rounds up to 360 degrees,
**     by testing for idmsf[0]=360 and setting idmsf[0-3] to zero.
**
**  Called:
**     iauD2tf      decompose days to hms
**
*/
```

```
void iauA2tf(int ndp, double angle, char *sign, int ihmsf[4])
/*
**  - - - - - - - -
**   i a u A 2 t f
**  - - - - - - - -
**
**  Decompose radians into hours, minutes, seconds, fraction.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     ndp     int     resolution (Note 1)
**     angle   double  angle in radians
**
**  Returned:
**     sign    char*   '+' or '-'
**     ihmsf   int[4]  hours, minutes, seconds, fraction
**
**  Notes:
**
**  1) The argument ndp is interpreted as follows:
**
**     ndp         resolution
**      :      ...0000 00 00
**     -7         1000 00 00
**     -6          100 00 00
**     -5           10 00 00
**     -4            1 00 00
**     -3            0 10 00
**     -2            0 01 00
**     -1            0 00 10
**      0            0 00 01
**      1            0 00 00.1
**      2            0 00 00.01
**      3            0 00 00.001
**      :            0 00 00.000...
**
**  2) The largest positive useful value for ndp is determined by the
**     size of angle, the format of doubles on the target platform, and
**     the risk of overflowing ihmsf[3].  On a typical platform, for
**     angle up to 2pi, the available floating-point precision might
**     correspond to ndp=12.  However, the practical limit is typically
**     ndp=9, set by the capacity of a 32-bit int, or ndp=4 if int is
**     only 16 bits.
**
**  3) The absolute value of angle may exceed 2pi.  In cases where it
**     does not, it is up to the caller to test for and handle the
**     case where angle is very nearly 2pi and rounds up to 24 hours,
**     by testing for ihmsf[0]=24 and setting ihmsf[0-3] to zero.
**
**  Called:
**     iauD2tf      decompose days to hms
**
*/
```

```
void iauAb(double pnat[3], double v[3], double s, double bm1,
           double ppr[3])
/*
**  - - - - - -
**   i a u A b
**  - - - - - -
**
**  Apply aberration to transform natural direction into proper
**  direction.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**    pnat    double[3]   natural direction to the source (unit vector)
**    v       double[3]   observer barycentric velocity in units of c
**    s       double      distance between the Sun and the observer (au)
**    bm1     double      sqrt(1-|v|^2): reciprocal of Lorenz factor
**
**  Returned:
**    ppr     double[3]   proper direction to source (unit vector)
**
**  Notes:
**
**  1) The algorithm is based on Expr. (7.40) in the Explanatory
**     Supplement (Urban & Seidelmann 2013), but with the following
**     changes:
**
**     o  Rigorous rather than approximate normalization is applied.
**
**     o  The gravitational potential term from Expr. (7) in
**        Klioner (2003) is added, taking into account only the Sun's
**        contribution.  This has a maximum effect of about
**        0.4 microarcsecond.
**
**  2) In almost all cases, the maximum accuracy will be limited by the
**     supplied velocity.  For example, if the SOFA iauEpv00 function is
**     used, errors of up to 5 microarcseconds could occur.
**
**  References:
**
**     Urban, S. & Seidelmann, P. K. (eds), Explanatory Supplement to
**     the Astronomical Almanac, 3rd ed., University Science Books
**     (2013).
**
**     Klioner, Sergei A., "A practical relativistic model for micro-
**     arcsecond astrometry in space", Astr. J. 125, 1580-1597 (2003).
**
**  Called:
**     iauPdp       scalar product of two p-vectors
**
*/
```

```
void iauAe2hd (double az, double el, double phi,
               double *ha, double *dec)
/*
**  - - - - - - - - -
**   i a u A e 2 h d
**  - - - - - - - - -
**
**  Horizon to equatorial coordinates:  transform azimuth and altitude
**  to hour angle and declination.
**
**  Given:
**     az       double       azimuth
**     el       double       altitude (informally, elevation)
**     phi      double       site latitude
**
**  Returned:
**     ha       double       hour angle (local)
**     dec      double       declination
**
**  Notes:
**
**  1)  All the arguments are angles in radians.
**
**  2)  The sign convention for azimuth is north zero, east +pi/2.
**
**  3)  HA is returned in the range +/-pi.  Declination is returned in
**      the range +/-pi/2.
**
**  4)  The latitude phi is pi/2 minus the angle between the Earth's
**      rotation axis and the adopted zenith.  In many applications it
**      will be sufficient to use the published geodetic latitude of the
**      site.  In very precise (sub-arcsecond) applications, phi can be
**      corrected for polar motion.
**
**  5)  The azimuth az must be with respect to the rotational north pole,
**      as opposed to the ITRS pole, and an azimuth with respect to north
**      on a map of the Earth's surface will need to be adjusted for
**      polar motion if sub-arcsecond accuracy is required.
**
**  6)  Should the user wish to work with respect to the astronomical
**      zenith rather than the geodetic zenith, phi will need to be
**      adjusted for deflection of the vertical (often tens of
**      arcseconds), and the zero point of ha will also be affected.
**
**  7)  The transformation is the same as Ve = Ry(phi-pi/2)*Rz(pi)*Vh,
**      where Ve and Vh are lefthanded unit vectors in the (ha,dec) and
**      (az,el) systems respectively and Rz and Ry are rotations about
**      first the z-axis and then the y-axis.  (n.b. Rz(pi) simply
**      reverses the signs of the x and y components.)  For efficiency,
**      the algorithm is written out rather than calling other utility
**      functions.  For applications that require even greater
**      efficiency, additional savings are possible if constant terms
**      such as functions of latitude are computed once and for all.
**
**  8)  Again for efficiency, no range checking of arguments is carried
**      out.
**
**  Last revision:   2017 September 12
**
**  SOFA release 2023-10-11
**
**  Copyright (C) 2023 IAU SOFA Board.  See notes at end.
*/
{
   double sa, ca, se, ce, sp, cp, x, y, z, r;


/* Useful trig functions. */
   sa = sin(az);
   ca = cos(az);
   se = sin(el);
```

```
   ce = cos(el);
   sp = sin(phi);
   cp = cos(phi);

/* HA,Dec unit vector. */
   x = - ca*ce*sp + se*cp;
   y = - sa*ce;
   z = ca*ce*cp + se*sp;

/* To spherical. */
   r = sqrt(x*x + y*y);
   *ha = (r != 0.0) ? atan2(y,x) : 0.0;
   *dec = atan2(z,r);

/* Finished. */

/*----------------------------------------------------------------------
**
**  Copyright (C) 2023
**  Standards of Fundamental Astronomy Board
**  of the International Astronomical Union.
**
**  =====================
**  SOFA Software License
**  =====================
**
**  NOTICE TO USER:
**
**  BY USING THIS SOFTWARE YOU ACCEPT THE FOLLOWING SIX TERMS AND
**  CONDITIONS WHICH APPLY TO ITS USE.
**
**  1. The Software is owned by the IAU SOFA Board ("SOFA").
**
**  2. Permission is granted to anyone to use the SOFA software for any
**     purpose, including commercial applications, free of charge and
**     without payment of royalties, subject to the conditions and
**     restrictions listed below.
**
**  3. You (the user) may copy and distribute SOFA source code to others,
**     and use and adapt its code and algorithms in your own software,
**     on a world-wide, royalty-free basis.  That portion of your
**     distribution that does not consist of intact and unchanged copies
**     of SOFA source code files is a "derived work" that must comply
**     with the following requirements:
**
**     a) Your work shall be marked or carry a statement that it
**        (i) uses routines and computations derived by you from
**        software provided by SOFA under license to you; and
**        (ii) does not itself constitute software provided by and/or
**        endorsed by SOFA.
**
**     b) The source code of your derived work must contain descriptions
**        of how the derived work is based upon, contains and/or differs
**        from the original SOFA software.
**
**     c) The names of all routines in your derived work shall not
**        include the prefix "iau" or "sofa" or trivial modifications
**        thereof such as changes of case.
**
**     d) The origin of the SOFA components of your derived work must
**        not be misrepresented;  you must not claim that you wrote the
**        original software, nor file a patent application for SOFA
**        software or algorithms embedded in the SOFA software.
**
**     e) These requirements must be reproduced intact in any source
**        distribution and shall apply to anyone to whom you have
**        granted a further right to modify the source code of your
**        derived work.
**
**     Note that, as originally distributed, the SOFA software is
**     intended to be a definitive implementation of the IAU standards,
**     and consequently third-party modifications are discouraged.  All
**     variations, no matter how minor, must be explicitly marked as
```

```
**     such, as explained above.
**
**  4. You shall not cause the SOFA software to be brought into
**     disrepute, either by misuse, or use for inappropriate tasks, or
**     by inappropriate modification.
**
**  5. The SOFA software is provided "as is" and SOFA makes no warranty
**     as to its use or performance.   SOFA does not and cannot warrant
**     the performance or results which the user may obtain by using the
**     SOFA software.  SOFA makes no warranties, express or implied, as
**     to non-infringement of third party rights, merchantability, or
**     fitness for any particular purpose.  In no event will SOFA be
**     liable to the user for any consequential, incidental, or special
**     damages, including any lost profits or lost savings, even if a
**     SOFA representative has been advised of such damages, or for any
**     claim by any third party.
**
**  6. The provision of any version of the SOFA software under the terms
**     and conditions specified herein does not imply that future
**     versions will also be made available under the same terms and
**     conditions.
*
**  In any published work or commercial product which uses the SOFA
**  software directly, acknowledgement (see www.iausofa.org) is
**  appreciated.
**
**  Correspondence concerning SOFA software should be addressed as
**  follows:
**
**      By email:  sofa@ukho.gov.uk
**      By post:   IAU SOFA Center
**                 HM Nautical Almanac Office
**                 UK Hydrographic Office
**                 Admiralty Way, Taunton
**                 Somerset, TA1 2DN
**                 United Kingdom
**
**--------------------------------------------------------------------*/
}
```

```
int iauAf2a(char s, int ideg, int iamin, double asec, double *rad)
/*
**  - - - - - - - -
**   i a u A f 2 a
**  - - - - - - - -
**
**  Convert degrees, arcminutes, arcseconds to radians.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     s         char    sign:  '-' = negative, otherwise positive
**     ideg      int     degrees
**     iamin     int     arcminutes
**     asec      double  arcseconds
**
**  Returned:
**     rad       double  angle in radians
**
**  Returned (function value):
**               int     status:  0 = OK
**                                1 = ideg outside range 0-359
**                                2 = iamin outside range 0-59
**                                3 = asec outside range 0-59.999...
**
**  Notes:
**
**  1)  The result is computed even if any of the range checks fail.
**
**  2)  Negative ideg, iamin and/or asec produce a warning status, but
**      the absolute value is used in the conversion.
**
**  3)  If there are multiple errors, the status value reflects only the
**      first, the smallest taking precedence.
**
*/
```

```
double iauAnp(double a)
/*
**  - - - - - - - -
**   i a u A n p
**  - - - - - - - -
**
**  Normalize angle into the range 0 <= a < 2pi.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     a        double     angle (radians)
**
**  Returned (function value):
**              double     angle in range 0-2pi
**
*/
```

```
double iauAnpm(double a)
/*
**  - - - - - - - - 
**   i a u A n p m
**  - - - - - - - - 
**
**  Normalize angle into the range -pi <= a < +pi.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     a        double     angle (radians)
**
**  Returned (function value):
**              double     angle in range +/-pi
**
*/
```

```
void iauApcg(double date1, double date2,
             double ebpv[2][3], double ehp[3],
             iauASTROM *astrom)
/*
**  - - - - - - - -
**   i a u A p c g
**  - - - - - - - -
**
**  For a geocentric observer, prepare star-independent astrometry
**  parameters for transformations between ICRS and GCRS coordinates.
**  The Earth ephemeris is supplied by the caller.
**
**  The parameters produced by this function are required in the
**  parallax, light deflection and aberration parts of the astrometric
**  transformation chain.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1  double       TDB as a 2-part...
**     date2  double       ...Julian Date (Note 1)
**     ebpv   double[2][3] Earth barycentric pos/vel (au, au/day)
**     ehp    double[3]    Earth heliocentric position (au)
**
**  Returned:
**     astrom iauASTROM*   star-independent astrometry parameters:
**      pmt    double       PM time interval (SSB, Julian years)
**      eb     double[3]    SSB to observer (vector, au)
**      eh     double[3]    Sun to observer (unit vector)
**      em     double       distance from Sun to observer (au)
**      v      double[3]    barycentric observer velocity (vector, c)
**      bm1    double       sqrt(1-|v|^2): reciprocal of Lorenz factor
**      bpn    double[3][3] bias-precession-nutation matrix
**      along  double       unchanged
**      xpl    double       unchanged
**      ypl    double       unchanged
**      sphi   double       unchanged
**      cphi   double       unchanged
**      diurab double       unchanged
**      eral   double       unchanged
**      refa   double       unchanged
**      refb   double       unchanged
**
**  Notes:
**
**  1) The TDB date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TDB)=2450123.7 could be expressed in any of these ways, among
**     others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in cases
**     where the loss of several decimal digits of resolution is
**     acceptable.  The J2000 method is best matched to the way the
**     argument is handled internally and will deliver the optimum
**     resolution.  The MJD method and the date & time methods are both
**     good compromises between resolution and convenience.  For most
**     applications of this function the choice will not be at all
**     critical.
**
**     TT can be used instead of TDB without any significant impact on
**     accuracy.
```

```
**
**  2) All the vectors are with respect to BCRS axes.
**
**  3) This is one of several functions that inserts into the astrom
**     structure star-independent parameters needed for the chain of
**     astrometric transformations ICRS <-> GCRS <-> CIRS <-> observed.
**
**     The various functions support different classes of observer and
**     portions of the transformation chain:
**
**          functions         observer        transformation
**
**       iauApcg iauApcg13    geocentric      ICRS <-> GCRS
**       iauApci iauApci13    terrestrial     ICRS <-> CIRS
**       iauApco iauApco13    terrestrial     ICRS <-> observed
**       iauApcs iauApcs13    space           ICRS <-> GCRS
**       iauAper iauAper13    terrestrial     update Earth rotation
**       iauApio iauApio13    terrestrial     CIRS <-> observed
**
**     Those with names ending in "13" use contemporary SOFA models to
**     compute the various ephemerides.  The others accept ephemerides
**     supplied by the caller.
**
**     The transformation from ICRS to GCRS covers space motion,
**     parallax, light deflection, and aberration.  From GCRS to CIRS
**     comprises frame bias and precession-nutation.  From CIRS to
**     observed takes account of Earth rotation, polar motion, diurnal
**     aberration and parallax (unless subsumed into the ICRS <-> GCRS
**     transformation), and atmospheric refraction.
**
**  4) The context structure astrom produced by this function is used by
**     iauAtciq* and iauAticq*.
**
**  Called:
**     iauApcs      astrometry parameters, ICRS-GCRS, space observer
**
*/
```

```
void iauApcg13(double date1, double date2, iauASTROM *astrom)
/*
**  - - - - - - - - - -
**   i a u A p c g 1 3
**  - - - - - - - - - -
**
**  For a geocentric observer, prepare star-independent astrometry
**  parameters for transformations between ICRS and GCRS coordinates.
**  The caller supplies the date, and SOFA models are used to predict
**  the Earth ephemeris.
**
**  The parameters produced by this function are required in the
**  parallax, light deflection and aberration parts of the astrometric
**  transformation chain.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1  double     TDB as a 2-part...
**     date2  double     ...Julian Date (Note 1)
**
**  Returned:
**     astrom iauASTROM* star-independent astrometry parameters:
**      pmt    double       PM time interval (SSB, Julian years)
**      eb     double[3]    SSB to observer (vector, au)
**      eh     double[3]    Sun to observer (unit vector)
**      em     double       distance from Sun to observer (au)
**      v      double[3]    barycentric observer velocity (vector, c)
**      bm1    double       sqrt(1-|v|^2): reciprocal of Lorenz factor
**      bpn    double[3][3] bias-precession-nutation matrix
**      along  double       unchanged
**      xpl    double       unchanged
**      ypl    double       unchanged
**      sphi   double       unchanged
**      cphi   double       unchanged
**      diurab double       unchanged
**      eral   double       unchanged
**      refa   double       unchanged
**      refb   double       unchanged
**
**  Notes:
**
**  1) The TDB date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TDB)=2450123.7 could be expressed in any of these ways, among
**     others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in cases
**     where the loss of several decimal digits of resolution is
**     acceptable.  The J2000 method is best matched to the way the
**     argument is handled internally and will deliver the optimum
**     resolution.  The MJD method and the date & time methods are both
**     good compromises between resolution and convenience.  For most
**     applications of this function the choice will not be at all
**     critical.
**
**     TT can be used instead of TDB without any significant impact on
**     accuracy.
**
**  2) All the vectors are with respect to BCRS axes.
**
```

```
**  3) In cases where the caller wishes to supply his own Earth
**     ephemeris, the function iauApcg can be used instead of the present
**     function.
**
**  4) This is one of several functions that inserts into the astrom
**     structure star-independent parameters needed for the chain of
**     astrometric transformations ICRS <-> GCRS <-> CIRS <-> observed.
**
**     The various functions support different classes of observer and
**     portions of the transformation chain:
**
**          functions         observer        transformation
**
**       iauApcg iauApcg13    geocentric      ICRS <-> GCRS
**       iauApci iauApci13    terrestrial     ICRS <-> CIRS
**       iauApco iauApco13    terrestrial     ICRS <-> observed
**       iauApcs iauApcs13    space           ICRS <-> GCRS
**       iauAper iauAper13    terrestrial     update Earth rotation
**       iauApio iauApio13    terrestrial     CIRS <-> observed
**
**     Those with names ending in "13" use contemporary SOFA models to
**     compute the various ephemerides.  The others accept ephemerides
**     supplied by the caller.
**
**     The transformation from ICRS to GCRS covers space motion,
**     parallax, light deflection, and aberration.  From GCRS to CIRS
**     comprises frame bias and precession-nutation.  From CIRS to
**     observed takes account of Earth rotation, polar motion, diurnal
**     aberration and parallax (unless subsumed into the ICRS <-> GCRS
**     transformation), and atmospheric refraction.
**
**  5) The context structure astrom produced by this function is used by
**     iauAtciq* and iauAticq*.
**
**  Called:
**     iauEpv00     Earth position and velocity
**     iauApcg      astrometry parameters, ICRS-GCRS, geocenter
**
*/
```

```
void iauApci(double date1, double date2,
             double ebpv[2][3], double ehp[3],
             double x, double y, double s,
             iauASTROM *astrom)
/*
**  - - - - - - - -
**   i a u A p c i
**  - - - - - - - -
**
**  For a terrestrial observer, prepare star-independent astrometry
**  parameters for transformations between ICRS and geocentric CIRS
**  coordinates.  The Earth ephemeris and CIP/CIO are supplied by the
**  caller.
**
**  The parameters produced by this function are required in the
**  parallax, light deflection, aberration, and bias-precession-nutation
**  parts of the astrometric transformation chain.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1  double       TDB as a 2-part...
**     date2  double       ...Julian Date (Note 1)
**     ebpv   double[2][3] Earth barycentric position/velocity (au, au/day)
**     ehp    double[3]    Earth heliocentric position (au)
**     x,y    double       CIP X,Y (components of unit vector)
**     s      double       the CIO locator s (radians)
**
**  Returned:
**     astrom iauASTROM*   star-independent astrometry parameters:
**      pmt    double       PM time interval (SSB, Julian years)
**      eb     double[3]    SSB to observer (vector, au)
**      eh     double[3]    Sun to observer (unit vector)
**      em     double       distance from Sun to observer (au)
**      v      double[3]    barycentric observer velocity (vector, c)
**      bm1    double       sqrt(1-|v|^2): reciprocal of Lorenz factor
**      bpn    double[3][3] bias-precession-nutation matrix
**      along  double       unchanged
**      xpl    double       unchanged
**      ypl    double       unchanged
**      sphi   double       unchanged
**      cphi   double       unchanged
**      diurab double       unchanged
**      eral   double       unchanged
**      refa   double       unchanged
**      refb   double       unchanged
**
**  Notes:
**
**  1) The TDB date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TDB)=2450123.7 could be expressed in any of these ways, among
**     others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in cases
**     where the loss of several decimal digits of resolution is
**     acceptable.  The J2000 method is best matched to the way the
**     argument is handled internally and will deliver the optimum
**     resolution.  The MJD method and the date & time methods are both
**     good compromises between resolution and convenience.  For most
**     applications of this function the choice will not be at all
```

```
**     critical.
**
**     TT can be used instead of TDB without any significant impact on
**     accuracy.
**
**  2) All the vectors are with respect to BCRS axes.
**
**  3) In cases where the caller does not wish to provide the Earth
**     ephemeris and CIP/CIO, the function iauApci13 can be used instead
**     of the present function.  This computes the required quantities
**     using other SOFA functions.
**
**  4) This is one of several functions that inserts into the astrom
**     structure star-independent parameters needed for the chain of
**     astrometric transformations ICRS <-> GCRS <-> CIRS <-> observed.
**
**     The various functions support different classes of observer and
**     portions of the transformation chain:
**
**          functions         observer        transformation
**
**       iauApcg iauApcg13    geocentric      ICRS <-> GCRS
**       iauApci iauApci13    terrestrial     ICRS <-> CIRS
**       iauApco iauApco13    terrestrial     ICRS <-> observed
**       iauApcs iauApcs13    space           ICRS <-> GCRS
**       iauAper iauAper13    terrestrial     update Earth rotation
**       iauApio iauApio13    terrestrial     CIRS <-> observed
**
**     Those with names ending in "13" use contemporary SOFA models to
**     compute the various ephemerides.  The others accept ephemerides
**     supplied by the caller.
**
**     The transformation from ICRS to GCRS covers space motion,
**     parallax, light deflection, and aberration.  From GCRS to CIRS
**     comprises frame bias and precession-nutation.  From CIRS to
**     observed takes account of Earth rotation, polar motion, diurnal
**     aberration and parallax (unless subsumed into the ICRS <-> GCRS
**     transformation), and atmospheric refraction.
**
**  5) The context structure astrom produced by this function is used by
**     iauAtciq* and iauAticq*.
**
**  Called:
**     iauApcg      astrometry parameters, ICRS-GCRS, geocenter
**     iauC2ixys    celestial-to-intermediate matrix, given X,Y and s
**
*/
```

```
void iauApci13(double date1, double date2,
               iauASTROM *astrom, double *eo)
/*
**  - - - - - - - - - -
**   i a u A p c i 1 3
**  - - - - - - - - - -
**
**  For a terrestrial observer, prepare star-independent astrometry
**  parameters for transformations between ICRS and geocentric CIRS
**  coordinates.  The caller supplies the date, and SOFA models are used
**  to predict the Earth ephemeris and CIP/CIO.
**
**  The parameters produced by this function are required in the
**  parallax, light deflection, aberration, and bias-precession-nutation
**  parts of the astrometric transformation chain.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1  double      TDB as a 2-part...
**     date2  double      ...Julian Date (Note 1)
**
**  Returned:
**     astrom iauASTROM*  star-independent astrometry parameters:
**      pmt    double       PM time interval (SSB, Julian years)
**      eb     double[3]    SSB to observer (vector, au)
**      eh     double[3]    Sun to observer (unit vector)
**      em     double       distance from Sun to observer (au)
**      v      double[3]    barycentric observer velocity (vector, c)
**      bm1    double       sqrt(1-|v|^2): reciprocal of Lorenz factor
**      bpn    double[3][3] bias-precession-nutation matrix
**      along  double       unchanged
**      xpl    double       unchanged
**      ypl    double       unchanged
**      sphi   double       unchanged
**      cphi   double       unchanged
**      diurab double       unchanged
**      eral   double       unchanged
**      refa   double       unchanged
**      refb   double       unchanged
**     eo     double*     equation of the origins (ERA-GST, radians)
**
**  Notes:
**
**  1) The TDB date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TDB)=2450123.7 could be expressed in any of these ways, among
**     others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in cases
**     where the loss of several decimal digits of resolution is
**     acceptable.  The J2000 method is best matched to the way the
**     argument is handled internally and will deliver the optimum
**     resolution.  The MJD method and the date & time methods are both
**     good compromises between resolution and convenience.  For most
**     applications of this function the choice will not be at all
**     critical.
**
**     TT can be used instead of TDB without any significant impact on
**     accuracy.
**
```

```
**  2) All the vectors are with respect to BCRS axes.
**
**  3) In cases where the caller wishes to supply his own Earth
**     ephemeris and CIP/CIO, the function iauApci can be used instead
**     of the present function.
**
**  4) This is one of several functions that inserts into the astrom
**     structure star-independent parameters needed for the chain of
**     astrometric transformations ICRS <-> GCRS <-> CIRS <-> observed.
**
**     The various functions support different classes of observer and
**     portions of the transformation chain:
**
**          functions         observer        transformation
**
**       iauApcg iauApcg13    geocentric      ICRS <-> GCRS
**       iauApci iauApci13    terrestrial     ICRS <-> CIRS
**       iauApco iauApco13    terrestrial     ICRS <-> observed
**       iauApcs iauApcs13    space           ICRS <-> GCRS
**       iauAper iauAper13    terrestrial     update Earth rotation
**       iauApio iauApio13    terrestrial     CIRS <-> observed
**
**     Those with names ending in "13" use contemporary SOFA models to
**     compute the various ephemerides.  The others accept ephemerides
**     supplied by the caller.
**
**     The transformation from ICRS to GCRS covers space motion,
**     parallax, light deflection, and aberration.  From GCRS to CIRS
**     comprises frame bias and precession-nutation.  From CIRS to
**     observed takes account of Earth rotation, polar motion, diurnal
**     aberration and parallax (unless subsumed into the ICRS <-> GCRS
**     transformation), and atmospheric refraction.
**
**  5) The context structure astrom produced by this function is used by
**     iauAtciq* and iauAticq*.
**
**  Called:
**     iauEpv00     Earth position and velocity
**     iauPnm06a    classical NPB matrix, IAU 2006/2000A
**     iauBpn2xy    extract CIP X,Y coordinates from NPB matrix
**     iauS06       the CIO locator s, given X,Y, IAU 2006
**     iauApci      astrometry parameters, ICRS-CIRS
**     iauEors      equation of the origins, given NPB matrix and s
**
*/
```

```
void iauApco(double date1, double date2,
             double ebpv[2][3], double ehp[3],
             double x, double y, double s, double theta,
             double elong, double phi, double hm,
             double xp, double yp, double sp,
             double refa, double refb,
             iauASTROM *astrom)
/*
**  - - - - - - - -
**   i a u A p c o
**  - - - - - - - -
**
**  For a terrestrial observer, prepare star-independent astrometry
**  parameters for transformations between ICRS and observed
**  coordinates.  The caller supplies the Earth ephemeris, the Earth
**  rotation information and the refraction constants as well as the
**  site coordinates.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1  double       TDB as a 2-part...
**     date2  double       ...Julian Date (Note 1)
**     ebpv   double[2][3] Earth barycentric PV (au, au/day, Note 2)
**     ehp    double[3]    Earth heliocentric P (au, Note 2)
**     x,y    double       CIP X,Y (components of unit vector)
**     s      double       the CIO locator s (radians)
**     theta  double       Earth rotation angle (radians)
**     elong  double       longitude (radians, east +ve, Note 3)
**     phi    double       latitude (geodetic, radians, Note 3)
**     hm     double       height above ellipsoid (m, geodetic, Note 3)
**     xp,yp  double       polar motion coordinates (radians, Note 4)
**     sp     double       the TIO locator s' (radians, Note 4)
**     refa   double       refraction constant A (radians, Note 5)
**     refb   double       refraction constant B (radians, Note 5)
**
**  Returned:
**     astrom iauASTROM*   star-independent astrometry parameters:
**      pmt    double       PM time interval (SSB, Julian years)
**      eb     double[3]    SSB to observer (vector, au)
**      eh     double[3]    Sun to observer (unit vector)
**      em     double       distance from Sun to observer (au)
**      v      double[3]    barycentric observer velocity (vector, c)
**      bm1    double       sqrt(1-|v|^2): reciprocal of Lorenz factor
**      bpn    double[3][3] bias-precession-nutation matrix
**      along  double       adjusted longitude (radians)
**      xpl    double       polar motion xp wrt local meridian (radians)
**      ypl    double       polar motion yp wrt local meridian (radians)
**      sphi   double       sine of geodetic latitude
**      cphi   double       cosine of geodetic latitude
**      diurab double       magnitude of diurnal aberration vector
**      eral   double       "local" Earth rotation angle (radians)
**      refa   double       refraction constant A (radians)
**      refb   double       refraction constant B (radians)
**
**  Notes:
**
**  1) The TDB date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TDB)=2450123.7 could be expressed in any of these ways, among
**     others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
```

```
**
**     The JD method is the most natural and convenient to use in cases
**     where the loss of several decimal digits of resolution is
**     acceptable.  The J2000 method is best matched to the way the
**     argument is handled internally and will deliver the optimum
**     resolution.  The MJD method and the date & time methods are both
**     good compromises between resolution and convenience.  For most
**     applications of this function the choice will not be at all
**     critical.
**
**     TT can be used instead of TDB without any significant impact on
**     accuracy.
**
**  2) The vectors eb, eh, and all the astrom vectors, are with respect
**     to BCRS axes.
**
**  3) The geographical coordinates are with respect to the WGS84
**     reference ellipsoid.  TAKE CARE WITH THE LONGITUDE SIGN
**     CONVENTION:  the longitude required by the present function is
**     right-handed, i.e. east-positive, in accordance with geographical
**     convention.
**
**     The adjusted longitude stored in the astrom array takes into
**     account the TIO locator and polar motion.
**
**  4) xp and yp are the coordinates (in radians) of the Celestial
**     Intermediate Pole with respect to the International Terrestrial
**     Reference System (see IERS Conventions), measured along the
**     meridians 0 and 90 deg west respectively.  sp is the TIO locator
**     s', in radians, which positions the Terrestrial Intermediate
**     Origin on the equator.  For many applications, xp, yp and
**     (especially) sp can be set to zero.
**
**     Internally, the polar motion is stored in a form rotated onto the
**     local meridian.
**
**  5) The refraction constants refa and refb are for use in a
**     dZ = A*tan(Z)+B*tan^3(Z) model, where Z is the observed
**     (i.e. refracted) zenith distance and dZ is the amount of
**     refraction.
**
**  6) It is advisable to take great care with units, as even unlikely
**     values of the input parameters are accepted and processed in
**     accordance with the models used.
**
**  7) In cases where the caller does not wish to provide the Earth
**     Ephemeris, the Earth rotation information and refraction
**     constants, the function iauApco13 can be used instead of the
**     present function.  This starts from UTC and weather readings etc.
**     and computes suitable values using other SOFA functions.
**
**  8) This is one of several functions that inserts into the astrom
**     structure star-independent parameters needed for the chain of
**     astrometric transformations ICRS <-> GCRS <-> CIRS <-> observed.
**
**     The various functions support different classes of observer and
**     portions of the transformation chain:
**
**          functions         observer        transformation
**
**       iauApcg iauApcg13    geocentric      ICRS <-> GCRS
**       iauApci iauApci13    terrestrial     ICRS <-> CIRS
**       iauApco iauApco13    terrestrial     ICRS <-> observed
**       iauApcs iauApcs13    space           ICRS <-> GCRS
**       iauAper iauAper13    terrestrial     update Earth rotation
**       iauApio iauApio13    terrestrial     CIRS <-> observed
**
**     Those with names ending in "13" use contemporary SOFA models to
**     compute the various ephemerides.  The others accept ephemerides
**     supplied by the caller.
**
**     The transformation from ICRS to GCRS covers space motion,
**     parallax, light deflection, and aberration.  From GCRS to CIRS
```

```
**     comprises frame bias and precession-nutation.  From CIRS to
**     observed takes account of Earth rotation, polar motion, diurnal
**     aberration and parallax (unless subsumed into the ICRS <-> GCRS
**     transformation), and atmospheric refraction.
**
**  9) The context structure astrom produced by this function is used by
**     iauAtioq, iauAtoiq, iauAtciq* and iauAticq*.
**
**  Called:
**     iauIr        initialize r-matrix to identity
**     iauRz        rotate around Z-axis
**     iauRy        rotate around Y-axis
**     iauRx        rotate around X-axis
**     iauAnpm      normalize angle into range +/- pi
**     iauC2ixys    celestial-to-intermediate matrix, given X,Y and s
**     iauPvtob     position/velocity of terrestrial station
**     iauTrxpv     product of transpose of r-matrix and pv-vector
**     iauApcs      astrometry parameters, ICRS-GCRS, space observer
**     iauCr        copy r-matrix
**
*/
```

```
int iauApco13(double utc1, double utc2, double dut1,
              double elong, double phi, double hm, double xp, double yp,
              double phpa, double tc, double rh, double wl,
              iauASTROM *astrom, double *eo)
/*
**  - - - - - - - - - -
**   i a u A p c o 1 3
**  - - - - - - - - - -
**
**  For a terrestrial observer, prepare star-independent astrometry
**  parameters for transformations between ICRS and observed
**  coordinates.  The caller supplies UTC, site coordinates, ambient air
**  conditions and observing wavelength, and SOFA models are used to
**  obtain the Earth ephemeris, CIP/CIO and refraction constants.
**
**  The parameters produced by this function are required in the
**  parallax, light deflection, aberration, and bias-precession-nutation
**  parts of the ICRS/CIRS transformations.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     utc1   double     UTC as a 2-part...
**     utc2   double     ...quasi Julian Date (Notes 1,2)
**     dut1   double     UT1-UTC (seconds, Note 3)
**     elong  double     longitude (radians, east +ve, Note 4)
**     phi    double     latitude (geodetic, radians, Note 4)
**     hm     double     height above ellipsoid (m, geodetic, Notes 4,6)
**     xp,yp  double     polar motion coordinates (radians, Note 5)
**     phpa   double     pressure at the observer (hPa = mB, Note 6)
**     tc     double     ambient temperature at the observer (deg C)
**     rh     double     relative humidity at the observer (range 0-1)
**     wl     double     wavelength (micrometers, Note 7)
**
**  Returned:
**     astrom iauASTROM* star-independent astrometry parameters:
**      pmt    double       PM time interval (SSB, Julian years)
**      eb     double[3]    SSB to observer (vector, au)
**      eh     double[3]    Sun to observer (unit vector)
**      em     double       distance from Sun to observer (au)
**      v      double[3]    barycentric observer velocity (vector, c)
**      bm1    double       sqrt(1-|v|^2): reciprocal of Lorenz factor
**      bpn    double[3][3] bias-precession-nutation matrix
**      along  double       longitude + s' (radians)
**      xpl    double       polar motion xp wrt local meridian (radians)
**      ypl    double       polar motion yp wrt local meridian (radians)
**      sphi   double       sine of geodetic latitude
**      cphi   double       cosine of geodetic latitude
**      diurab double       magnitude of diurnal aberration vector
**      eral   double       "local" Earth rotation angle (radians)
**      refa   double       refraction constant A (radians)
**      refb   double       refraction constant B (radians)
**     eo     double*    equation of the origins (ERA-GST, radians)
**
**  Returned (function value):
**            int        status: +1 = dubious year (Note 2)
**                                0 = OK
**                               -1 = unacceptable date
**
**  Notes:
**
**  1)  utc1+utc2 is quasi Julian Date (see Note 2), apportioned in any
**      convenient way between the two arguments, for example where utc1
**      is the Julian Day Number and utc2 is the fraction of a day.
**
**      However, JD cannot unambiguously represent UTC during a leap
**      second unless special measures are taken.  The convention in the
**      present function is that the JD day represents UTC days whether
```

```
**       the length is 86399, 86400 or 86401 SI seconds.
**
**       Applications should use the function iauDtf2d to convert from
**       calendar date and time of day into 2-part quasi Julian Date, as
**       it implements the leap-second-ambiguity convention just
**       described.
**
**   2)  The warning status "dubious year" flags UTCs that predate the
**       introduction of the time scale or that are too far in the
**       future to be trusted.  See iauDat for further details.
**
**   3)  UT1-UTC is tabulated in IERS bulletins.  It increases by exactly
**       one second at the end of each positive UTC leap second,
**       introduced in order to keep UT1-UTC within +/- 0.9s.  n.b. This
**       practice is under review, and in the future UT1-UTC may grow
**       essentially without limit.
**
**   4)  The geographical coordinates are with respect to the WGS84
**       reference ellipsoid.  TAKE CARE WITH THE LONGITUDE SIGN:  the
**       longitude required by the present function is east-positive
**       (i.e. right-handed), in accordance with geographical convention.
**
**   5)  The polar motion xp,yp can be obtained from IERS bulletins.  The
**       values are the coordinates (in radians) of the Celestial
**       Intermediate Pole with respect to the International Terrestrial
**       Reference System (see IERS Conventions 2003), measured along the
**       meridians 0 and 90 deg west respectively.  For many
**       applications, xp and yp can be set to zero.
**
**       Internally, the polar motion is stored in a form rotated onto
**       the local meridian.
**
**   6)  If hm, the height above the ellipsoid of the observing station
**       in meters, is not known but phpa, the pressure in hPa (=mB), is
**       available, an adequate estimate of hm can be obtained from the
**       expression
**
**             hm = -29.3 * tsl * log ( phpa / 1013.25 );
**
**       where tsl is the approximate sea-level air temperature in K
**       (See Astrophysical Quantities, C.W.Allen, 3rd edition, section
**       52).  Similarly, if the pressure phpa is not known, it can be
**       estimated from the height of the observing station, hm, as
**       follows:
**
**             phpa = 1013.25 * exp ( -hm / ( 29.3 * tsl ) );
**
**       Note, however, that the refraction is nearly proportional to
**       the pressure and that an accurate phpa value is important for
**       precise work.
**
**   7)  The argument wl specifies the observing wavelength in
**       micrometers.  The transition from optical to radio is assumed to
**       occur at 100 micrometers (about 3000 GHz).
**
**   8)  It is advisable to take great care with units, as even unlikely
**       values of the input parameters are accepted and processed in
**       accordance with the models used.
**
**   9)  In cases where the caller wishes to supply his own Earth
**       ephemeris, Earth rotation information and refraction constants,
**       the function iauApco can be used instead of the present function.
**
**   10) This is one of several functions that inserts into the astrom
**       structure star-independent parameters needed for the chain of
**       astrometric transformations ICRS <-> GCRS <-> CIRS <-> observed.
**
**       The various functions support different classes of observer and
**       portions of the transformation chain:
**
**           functions         observer        transformation
**
**        iauApcg iauApcg13    geocentric      ICRS <-> GCRS
```

```
**        iauApci iauApci13    terrestrial     ICRS <-> CIRS
**        iauApco iauApco13    terrestrial     ICRS <-> observed
**        iauApcs iauApcs13    space           ICRS <-> GCRS
**        iauAper iauAper13    terrestrial     update Earth rotation
**        iauApio iauApio13    terrestrial     CIRS <-> observed
**
**       Those with names ending in "13" use contemporary SOFA models to
**       compute the various ephemerides.  The others accept ephemerides
**       supplied by the caller.
**
**       The transformation from ICRS to GCRS covers space motion,
**       parallax, light deflection, and aberration.  From GCRS to CIRS
**       comprises frame bias and precession-nutation.  From CIRS to
**       observed takes account of Earth rotation, polar motion, diurnal
**       aberration and parallax (unless subsumed into the ICRS <-> GCRS
**       transformation), and atmospheric refraction.
**
**   11) The context structure astrom produced by this function is used
**       by iauAtioq, iauAtoiq, iauAtciq* and iauAticq*.
**
**   Called:
**      iauUtctai    UTC to TAI
**      iauTaitt     TAI to TT
**      iauUtcut1    UTC to UT1
**      iauEpv00     Earth position and velocity
**      iauPnm06a    classical NPB matrix, IAU 2006/2000A
**      iauBpn2xy    extract CIP X,Y coordinates from NPB matrix
**      iauS06       the CIO locator s, given X,Y, IAU 2006
**      iauEra00     Earth rotation angle, IAU 2000
**      iauSp00      the TIO locator s', IERS 2000
**      iauRefco     refraction constants for given ambient conditions
**      iauApco      astrometry parameters, ICRS-observed
**      iauEors      equation of the origins, given NPB matrix and s
**
*/
```

```
void iauApcs(double date1, double date2, double pv[2][3],
             double ebpv[2][3], double ehp[3],
             iauASTROM *astrom)
/*
**  - - - - - - - -
**   i a u A p c s
**  - - - - - - - -
**
**  For an observer whose geocentric position and velocity are known,
**  prepare star-independent astrometry parameters for transformations
**  between ICRS and GCRS.  The Earth ephemeris is supplied by the
**  caller.
**
**  The parameters produced by this function are required in the space
**  motion, parallax, light deflection and aberration parts of the
**  astrometric transformation chain.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1  double       TDB as a 2-part...
**     date2  double       ...Julian Date (Note 1)
**     pv     double[2][3] observer's geocentric pos/vel (m, m/s)
**     ebpv   double[2][3] Earth barycentric PV (au, au/day)
**     ehp    double[3]    Earth heliocentric P (au)
**
**  Returned:
**     astrom iauASTROM*   star-independent astrometry parameters:
**      pmt    double       PM time interval (SSB, Julian years)
**      eb     double[3]    SSB to observer (vector, au)
**      eh     double[3]    Sun to observer (unit vector)
**      em     double       distance from Sun to observer (au)
**      v      double[3]    barycentric observer velocity (vector, c)
**      bm1    double       sqrt(1-|v|^2): reciprocal of Lorenz factor
**      bpn    double[3][3] bias-precession-nutation matrix
**      along  double       unchanged
**      xpl    double       unchanged
**      ypl    double       unchanged
**      sphi   double       unchanged
**      cphi   double       unchanged
**      diurab double       unchanged
**      eral   double       unchanged
**      refa   double       unchanged
**      refb   double       unchanged
**
**  Notes:
**
**  1) The TDB date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TDB)=2450123.7 could be expressed in any of these ways, among
**     others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in cases
**     where the loss of several decimal digits of resolution is
**     acceptable.  The J2000 method is best matched to the way the
**     argument is handled internally and will deliver the optimum
**     resolution.  The MJD method and the date & time methods are both
**     good compromises between resolution and convenience.  For most
**     applications of this function the choice will not be at all
**     critical.
**
```

```
**     TT can be used instead of TDB without any significant impact on
**     accuracy.
**
**  2) All the vectors are with respect to BCRS axes.
**
**  3) Providing separate arguments for (i) the observer's geocentric
**     position and velocity and (ii) the Earth ephemeris is done for
**     convenience in the geocentric, terrestrial and Earth orbit cases.
**     For deep space applications it maybe more convenient to specify
**     zero geocentric position and velocity and to supply the
**     observer's position and velocity information directly instead of
**     with respect to the Earth.  However, note the different units:
**     m and m/s for the geocentric vectors, au and au/day for the
**     heliocentric and barycentric vectors.
**
**  4) In cases where the caller does not wish to provide the Earth
**     ephemeris, the function iauApcs13 can be used instead of the
**     present function.  This computes the Earth ephemeris using the
**     SOFA function iauEpv00.
**
**  5) This is one of several functions that inserts into the astrom
**     structure star-independent parameters needed for the chain of
**     astrometric transformations ICRS <-> GCRS <-> CIRS <-> observed.
**
**     The various functions support different classes of observer and
**     portions of the transformation chain:
**
**          functions         observer        transformation
**
**       iauApcg iauApcg13    geocentric      ICRS <-> GCRS
**       iauApci iauApci13    terrestrial     ICRS <-> CIRS
**       iauApco iauApco13    terrestrial     ICRS <-> observed
**       iauApcs iauApcs13    space           ICRS <-> GCRS
**       iauAper iauAper13    terrestrial     update Earth rotation
**       iauApio iauApio13    terrestrial     CIRS <-> observed
**
**     Those with names ending in "13" use contemporary SOFA models to
**     compute the various ephemerides.  The others accept ephemerides
**     supplied by the caller.
**
**     The transformation from ICRS to GCRS covers space motion,
**     parallax, light deflection, and aberration.  From GCRS to CIRS
**     comprises frame bias and precession-nutation.  From CIRS to
**     observed takes account of Earth rotation, polar motion, diurnal
**     aberration and parallax (unless subsumed into the ICRS <-> GCRS
**     transformation), and atmospheric refraction.
**
**  6) The context structure astrom produced by this function is used by
**     iauAtciq* and iauAticq*.
**
**  Called:
**     iauCp        copy p-vector
**     iauPm        modulus of p-vector
**     iauPn        decompose p-vector into modulus and direction
**     iauIr        initialize r-matrix to identity
**
*/
```

```
void iauApcs13(double date1, double date2, double pv[2][3],
               iauASTROM *astrom)
/*
**  - - - - - - - - - - -
**   i a u A p c s 1 3
**  - - - - - - - - - - -
**
**  For an observer whose geocentric position and velocity are known,
**  prepare star-independent astrometry parameters for transformations
**  between ICRS and GCRS.  The Earth ephemeris is from SOFA models.
**
**  The parameters produced by this function are required in the space
**  motion, parallax, light deflection and aberration parts of the
**  astrometric transformation chain.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1  double       TDB as a 2-part...
**     date2  double       ...Julian Date (Note 1)
**     pv     double[2][3] observer's geocentric pos/vel (Note 3)
**
**  Returned:
**     astrom iauASTROM*   star-independent astrometry parameters:
**      pmt    double       PM time interval (SSB, Julian years)
**      eb     double[3]    SSB to observer (vector, au)
**      eh     double[3]    Sun to observer (unit vector)
**      em     double       distance from Sun to observer (au)
**      v      double[3]    barycentric observer velocity (vector, c)
**      bm1    double       sqrt(1-|v|^2): reciprocal of Lorenz factor
**      bpn    double[3][3] bias-precession-nutation matrix
**      along  double       unchanged
**      xpl    double       unchanged
**      ypl    double       unchanged
**      sphi   double       unchanged
**      cphi   double       unchanged
**      diurab double       unchanged
**      eral   double       unchanged
**      refa   double       unchanged
**      refb   double       unchanged
**
**  Notes:
**
**  1) The TDB date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TDB)=2450123.7 could be expressed in any of these ways, among
**     others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in cases
**     where the loss of several decimal digits of resolution is
**     acceptable.  The J2000 method is best matched to the way the
**     argument is handled internally and will deliver the optimum
**     resolution.  The MJD method and the date & time methods are both
**     good compromises between resolution and convenience.  For most
**     applications of this function the choice will not be at all
**     critical.
**
**     TT can be used instead of TDB without any significant impact on
**     accuracy.
**
**  2) All the vectors are with respect to BCRS axes.
```

```
**
**  3) The observer's position and velocity pv are geocentric but with
**     respect to BCRS axes, and in units of m and m/s.  No assumptions
**     are made about proximity to the Earth, and the function can be
**     used for deep space applications as well as Earth orbit and
**     terrestrial.
**
**  4) In cases where the caller wishes to supply his own Earth
**     ephemeris, the function iauApcs can be used instead of the present
**     function.
**
**  5) This is one of several functions that inserts into the astrom
**     structure star-independent parameters needed for the chain of
**     astrometric transformations ICRS <-> GCRS <-> CIRS <-> observed.
**
**     The various functions support different classes of observer and
**     portions of the transformation chain:
**
**          functions         observer        transformation
**
**       iauApcg iauApcg13    geocentric      ICRS <-> GCRS
**       iauApci iauApci13    terrestrial     ICRS <-> CIRS
**       iauApco iauApco13    terrestrial     ICRS <-> observed
**       iauApcs iauApcs13    space           ICRS <-> GCRS
**       iauAper iauAper13    terrestrial     update Earth rotation
**       iauApio iauApio13    terrestrial     CIRS <-> observed
**
**     Those with names ending in "13" use contemporary SOFA models to
**     compute the various ephemerides.  The others accept ephemerides
**     supplied by the caller.
**
**     The transformation from ICRS to GCRS covers space motion,
**     parallax, light deflection, and aberration.  From GCRS to CIRS
**     comprises frame bias and precession-nutation.  From CIRS to
**     observed takes account of Earth rotation, polar motion, diurnal
**     aberration and parallax (unless subsumed into the ICRS <-> GCRS
**     transformation), and atmospheric refraction.
**
**  6) The context structure astrom produced by this function is used by
**     iauAtciq* and iauAticq*.
**
**  Called:
**     iauEpv00     Earth position and velocity
**     iauApcs      astrometry parameters, ICRS-GCRS, space observer
**
*/
```

```
void iauAper(double theta, iauASTROM *astrom)
/*
**  - - - - - - - -
**   i a u A p e r
**  - - - - - - - -
**
**  In the star-independent astrometry parameters, update only the
**  Earth rotation angle, supplied by the caller explicitly.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     theta   double      Earth rotation angle (radians, Note 2)
**     astrom  iauASTROM*  star-independent astrometry parameters:
**      pmt    double       not used
**      eb     double[3]    not used
**      eh     double[3]    not used
**      em     double       not used
**      v      double[3]    not used
**      bm1    double       not used
**      bpn    double[3][3] not used
**      along  double       longitude + s' (radians)
**      xpl    double       not used
**      ypl    double       not used
**      sphi   double       not used
**      cphi   double       not used
**      diurab double       not used
**      eral   double       not used
**      refa   double       not used
**      refb   double       not used
**
**  Returned:
**     astrom  iauASTROM*  star-independent astrometry parameters:
**      pmt    double       unchanged
**      eb     double[3]    unchanged
**      eh     double[3]    unchanged
**      em     double       unchanged
**      v      double[3]    unchanged
**      bm1    double       unchanged
**      bpn    double[3][3] unchanged
**      along  double       unchanged
**      xpl    double       unchanged
**      ypl    double       unchanged
**      sphi   double       unchanged
**      cphi   double       unchanged
**      diurab double       unchanged
**      eral   double       "local" Earth rotation angle (radians)
**      refa   double       unchanged
**      refb   double       unchanged
**
**  Notes:
**
**  1) This function exists to enable sidereal-tracking applications to
**     avoid wasteful recomputation of the bulk of the astrometry
**     parameters:  only the Earth rotation is updated.
**
**  2) For targets expressed as equinox based positions, such as
**     classical geocentric apparent (RA,Dec), the supplied theta can be
**     Greenwich apparent sidereal time rather than Earth rotation
**     angle.
**
**  3) The function iauAper13 can be used instead of the present
**     function, and starts from UT1 rather than ERA itself.
**
**  4) This is one of several functions that inserts into the astrom
**     structure star-independent parameters needed for the chain of
**     astrometric transformations ICRS <-> GCRS <-> CIRS <-> observed.
**
```

```
**     The various functions support different classes of observer and
**     portions of the transformation chain:
**
**          functions         observer        transformation
**
**       iauApcg iauApcg13    geocentric      ICRS <-> GCRS
**       iauApci iauApci13    terrestrial     ICRS <-> CIRS
**       iauApco iauApco13    terrestrial     ICRS <-> observed
**       iauApcs iauApcs13    space           ICRS <-> GCRS
**       iauAper iauAper13    terrestrial     update Earth rotation
**       iauApio iauApio13    terrestrial     CIRS <-> observed
**
**     Those with names ending in "13" use contemporary SOFA models to
**     compute the various ephemerides.  The others accept ephemerides
**     supplied by the caller.
**
**     The transformation from ICRS to GCRS covers space motion,
**     parallax, light deflection, and aberration.  From GCRS to CIRS
**     comprises frame bias and precession-nutation.  From CIRS to
**     observed takes account of Earth rotation, polar motion, diurnal
**     aberration and parallax (unless subsumed into the ICRS <-> GCRS
**     transformation), and atmospheric refraction.
**
*/
```

```
void iauAper13(double ut11, double ut12, iauASTROM *astrom)
/*
**  - - - - - - - - - -
**   i a u A p e r 1 3
**  - - - - - - - - - -
**
**  In the star-independent astrometry parameters, update only the
**  Earth rotation angle.  The caller provides UT1, (n.b. not UTC).
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     ut11    double      UT1 as a 2-part...
**     ut12    double      ...Julian Date (Note 1)
**     astrom  iauASTROM*  star-independent astrometry parameters:
**      pmt    double       not used
**      eb     double[3]    not used
**      eh     double[3]    not used
**      em     double       not used
**      v      double[3]    not used
**      bm1    double       not used
**      bpn    double[3][3] not used
**      along  double       longitude + s' (radians)
**      xpl    double       not used
**      ypl    double       not used
**      sphi   double       not used
**      cphi   double       not used
**      diurab double       not used
**      eral   double       not used
**      refa   double       not used
**      refb   double       not used
**
**  Returned:
**     astrom  iauASTROM*  star-independent astrometry parameters:
**      pmt    double       unchanged
**      eb     double[3]    unchanged
**      eh     double[3]    unchanged
**      em     double       unchanged
**      v      double[3]    unchanged
**      bm1    double       unchanged
**      bpn    double[3][3] unchanged
**      along  double       unchanged
**      xpl    double       unchanged
**      ypl    double       unchanged
**      sphi   double       unchanged
**      cphi   double       unchanged
**      diurab double       unchanged
**      eral   double       "local" Earth rotation angle (radians)
**      refa   double       unchanged
**      refb   double       unchanged
**
**  Notes:
**
**  1) The UT1 date (n.b. not UTC) ut11+ut12 is a Julian Date,
**     apportioned in any convenient way between the arguments ut11 and
**     ut12.  For example, JD(UT1)=2450123.7 could be expressed in any
**     of these ways, among others:
**
**            ut11           ut12
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in cases
**     where the loss of several decimal digits of resolution is
**     acceptable.  The J2000 and MJD methods are good compromises
```

```
**     between resolution and convenience.  The date & time method is
**     best matched to the algorithm used:  maximum precision is
**     delivered when the ut11 argument is for 0hrs UT1 on the day in
**     question and the ut12 argument lies in the range 0 to 1, or vice
**     versa.
**
**  2) If the caller wishes to provide the Earth rotation angle itself,
**     the function iauAper can be used instead.  One use of this
**     technique is to substitute Greenwich apparent sidereal time and
**     thereby to support equinox based transformations directly.
**
**  3) This is one of several functions that inserts into the astrom
**     structure star-independent parameters needed for the chain of
**     astrometric transformations ICRS <-> GCRS <-> CIRS <-> observed.
**
**     The various functions support different classes of observer and
**     portions of the transformation chain:
**
**          functions         observer        transformation
**
**       iauApcg iauApcg13    geocentric      ICRS <-> GCRS
**       iauApci iauApci13    terrestrial     ICRS <-> CIRS
**       iauApco iauApco13    terrestrial     ICRS <-> observed
**       iauApcs iauApcs13    space           ICRS <-> GCRS
**       iauAper iauAper13    terrestrial     update Earth rotation
**       iauApio iauApio13    terrestrial     CIRS <-> observed
**
**     Those with names ending in "13" use contemporary SOFA models to
**     compute the various ephemerides.  The others accept ephemerides
**     supplied by the caller.
**
**     The transformation from ICRS to GCRS covers space motion,
**     parallax, light deflection, and aberration.  From GCRS to CIRS
**     comprises frame bias and precession-nutation.  From CIRS to
**     observed takes account of Earth rotation, polar motion, diurnal
**     aberration and parallax (unless subsumed into the ICRS <-> GCRS
**     transformation), and atmospheric refraction.
**
**  Called:
**     iauAper      astrometry parameters: update ERA
**     iauEra00     Earth rotation angle, IAU 2000
**
*/
```

```
void iauApio(double sp, double theta,
             double elong, double phi, double hm, double xp, double yp,
             double refa, double refb,
             iauASTROM *astrom)
/*
**  - - - - - - - -
**   i a u A p i o
**  - - - - - - - -
**
**  For a terrestrial observer, prepare star-independent astrometry
**  parameters for transformations between CIRS and observed
**  coordinates.  The caller supplies the Earth orientation information
**  and the refraction constants as well as the site coordinates.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     sp     double      the TIO locator s' (radians, Note 1)
**     theta  double      Earth rotation angle (radians)
**     elong  double      longitude (radians, east +ve, Note 2)
**     phi    double      geodetic latitude (radians, Note 2)
**     hm     double      height above ellipsoid (m, geodetic Note 2)
**     xp,yp  double      polar motion coordinates (radians, Note 3)
**     refa   double      refraction constant A (radians, Note 4)
**     refb   double      refraction constant B (radians, Note 4)
**
**  Returned:
**     astrom iauASTROM*  star-independent astrometry parameters:
**      pmt    double       unchanged
**      eb     double[3]    unchanged
**      eh     double[3]    unchanged
**      em     double       unchanged
**      v      double[3]    unchanged
**      bm1    double       unchanged
**      bpn    double[3][3] unchanged
**      along  double       adjusted longitude (radians)
**      xpl    double       polar motion xp wrt local meridian (radians)
**      ypl    double       polar motion yp wrt local meridian (radians)
**      sphi   double       sine of geodetic latitude
**      cphi   double       cosine of geodetic latitude
**      diurab double       magnitude of diurnal aberration vector
**      eral   double       "local" Earth rotation angle (radians)
**      refa   double       refraction constant A (radians)
**      refb   double       refraction constant B (radians)
**
**  Notes:
**
**  1) sp, the TIO locator s', is a tiny quantity needed only by the
**     most precise applications.  It can either be set to zero or
**     predicted using the SOFA function iauSp00.
**
**  2) The geographical coordinates are with respect to the WGS84
**     reference ellipsoid.  TAKE CARE WITH THE LONGITUDE SIGN:  the
**     longitude required by the present function is east-positive
**     (i.e. right-handed), in accordance with geographical convention.
**
**  3) The polar motion xp,yp can be obtained from IERS bulletins.  The
**     values are the coordinates (in radians) of the Celestial
**     Intermediate Pole with respect to the International Terrestrial
**     Reference System (see IERS Conventions 2003), measured along the
**     meridians 0 and 90 deg west respectively.  For many applications,
**     xp and yp can be set to zero.
**
**     Internally, the polar motion is stored in a form rotated onto the
**     local meridian.
**
**  4) The refraction constants refa and refb are for use in a
**     dZ = A*tan(Z)+B*tan^3(Z) model, where Z is the observed
```

```
**     (i.e. refracted) zenith distance and dZ is the amount of
**     refraction.
**
**  5) It is advisable to take great care with units, as even unlikely
**     values of the input parameters are accepted and processed in
**     accordance with the models used.
**
**  6) In cases where the caller does not wish to provide the Earth
**     rotation information and refraction constants, the function
**     iauApio13 can be used instead of the present function.  This
**     starts from UTC and weather readings etc. and computes suitable
**     values using other SOFA functions.
**
**  7) This is one of several functions that inserts into the astrom
**     structure star-independent parameters needed for the chain of
**     astrometric transformations ICRS <-> GCRS <-> CIRS <-> observed.
**
**     The various functions support different classes of observer and
**     portions of the transformation chain:
**
**          functions         observer        transformation
**
**       iauApcg iauApcg13    geocentric      ICRS <-> GCRS
**       iauApci iauApci13    terrestrial     ICRS <-> CIRS
**       iauApco iauApco13    terrestrial     ICRS <-> observed
**       iauApcs iauApcs13    space           ICRS <-> GCRS
**       iauAper iauAper13    terrestrial     update Earth rotation
**       iauApio iauApio13    terrestrial     CIRS <-> observed
**
**     Those with names ending in "13" use contemporary SOFA models to
**     compute the various ephemerides.  The others accept ephemerides
**     supplied by the caller.
**
**     The transformation from ICRS to GCRS covers space motion,
**     parallax, light deflection, and aberration.  From GCRS to CIRS
**     comprises frame bias and precession-nutation.  From CIRS to
**     observed takes account of Earth rotation, polar motion, diurnal
**     aberration and parallax (unless subsumed into the ICRS <-> GCRS
**     transformation), and atmospheric refraction.
**
**  8) The context structure astrom produced by this function is used by
**     iauAtioq and iauAtoiq.
**
**  Called:
**     iauIr        initialize r-matrix to identity
**     iauRz        rotate around Z-axis
**     iauRy        rotate around Y-axis
**     iauRx        rotate around X-axis
**     iauAnpm      normalize angle into range +/- pi
**     iauPvtob     position/velocity of terrestrial station
**
*/
```

```
int iauApio13(double utc1, double utc2, double dut1,
              double elong, double phi, double hm, double xp, double yp,
              double phpa, double tc, double rh, double wl,
              iauASTROM *astrom)
/*
**  - - - - - - - - - -
**   i a u A p i o 1 3
**  - - - - - - - - - -
**
**  For a terrestrial observer, prepare star-independent astrometry
**  parameters for transformations between CIRS and observed
**  coordinates.  The caller supplies UTC, site coordinates, ambient air
**  conditions and observing wavelength.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     utc1   double      UTC as a 2-part...
**     utc2   double      ...quasi Julian Date (Notes 1,2)
**     dut1   double      UT1-UTC (seconds)
**     elong  double      longitude (radians, east +ve, Note 3)
**     phi    double      geodetic latitude (radians, Note 3)
**     hm     double      height above ellipsoid (m, geodetic Notes 4,6)
**     xp,yp  double      polar motion coordinates (radians, Note 5)
**     phpa   double      pressure at the observer (hPa = mB, Note 6)
**     tc     double      ambient temperature at the observer (deg C)
**     rh     double      relative humidity at the observer (range 0-1)
**     wl     double      wavelength (micrometers, Note 7)
**
**  Returned:
**     astrom iauASTROM*  star-independent astrometry parameters:
**      pmt    double       unchanged
**      eb     double[3]    unchanged
**      eh     double[3]    unchanged
**      em     double       unchanged
**      v      double[3]    unchanged
**      bm1    double       unchanged
**      bpn    double[3][3] unchanged
**      along  double       longitude + s' (radians)
**      xpl    double       polar motion xp wrt local meridian (radians)
**      ypl    double       polar motion yp wrt local meridian (radians)
**      sphi   double       sine of geodetic latitude
**      cphi   double       cosine of geodetic latitude
**      diurab double       magnitude of diurnal aberration vector
**      eral   double       "local" Earth rotation angle (radians)
**      refa   double       refraction constant A (radians)
**      refb   double       refraction constant B (radians)
**
**  Returned (function value):
**            int         status: +1 = dubious year (Note 2)
**                                 0 = OK
**                                -1 = unacceptable date
**
**  Notes:
**
**  1)  utc1+utc2 is quasi Julian Date (see Note 2), apportioned in any
**      convenient way between the two arguments, for example where utc1
**      is the Julian Day Number and utc2 is the fraction of a day.
**
**      However, JD cannot unambiguously represent UTC during a leap
**      second unless special measures are taken.  The convention in the
**      present function is that the JD day represents UTC days whether
**      the length is 86399, 86400 or 86401 SI seconds.
**
**      Applications should use the function iauDtf2d to convert from
**      calendar date and time of day into 2-part quasi Julian Date, as
**      it implements the leap-second-ambiguity convention just
**      described.
```

```
**
**  2)  The warning status "dubious year" flags UTCs that predate the
**      introduction of the time scale or that are too far in the future
**      to be trusted.  See iauDat for further details.
**
**  3)  UT1-UTC is tabulated in IERS bulletins.  It increases by exactly
**      one second at the end of each positive UTC leap second,
**      introduced in order to keep UT1-UTC within +/- 0.9s.  n.b. This
**      practice is under review, and in the future UT1-UTC may grow
**      essentially without limit.
**
**  4)  The geographical coordinates are with respect to the WGS84
**      reference ellipsoid.  TAKE CARE WITH THE LONGITUDE SIGN:  the
**      longitude required by the present function is east-positive
**      (i.e. right-handed), in accordance with geographical convention.
**
**  5)  The polar motion xp,yp can be obtained from IERS bulletins.  The
**      values are the coordinates (in radians) of the Celestial
**      Intermediate Pole with respect to the International Terrestrial
**      Reference System (see IERS Conventions 2003), measured along the
**      meridians 0 and 90 deg west respectively.  For many applications,
**      xp and yp can be set to zero.
**
**      Internally, the polar motion is stored in a form rotated onto
**      the local meridian.
**
**  6)  If hm, the height above the ellipsoid of the observing station
**      in meters, is not known but phpa, the pressure in hPa (=mB), is
**      available, an adequate estimate of hm can be obtained from the
**      expression
**
**            hm = -29.3 * tsl * log ( phpa / 1013.25 );
**
**      where tsl is the approximate sea-level air temperature in K
**      (See Astrophysical Quantities, C.W.Allen, 3rd edition, section
**      52).  Similarly, if the pressure phpa is not known, it can be
**      estimated from the height of the observing station, hm, as
**      follows:
**
**            phpa = 1013.25 * exp ( -hm / ( 29.3 * tsl ) );
**
**      Note, however, that the refraction is nearly proportional to the
**      pressure and that an accurate phpa value is important for
**      precise work.
**
**  7)  The argument wl specifies the observing wavelength in
**      micrometers.  The transition from optical to radio is assumed to
**      occur at 100 micrometers (about 3000 GHz).
**
**  8)  It is advisable to take great care with units, as even unlikely
**      values of the input parameters are accepted and processed in
**      accordance with the models used.
**
**  9)  In cases where the caller wishes to supply his own Earth
**      rotation information and refraction constants, the function
**      iauApc can be used instead of the present function.
**
**  10) This is one of several functions that inserts into the astrom
**      structure star-independent parameters needed for the chain of
**      astrometric transformations ICRS <-> GCRS <-> CIRS <-> observed.
**
**      The various functions support different classes of observer and
**      portions of the transformation chain:
**
**          functions         observer        transformation
**
**       iauApcg iauApcg13    geocentric      ICRS <-> GCRS
**       iauApci iauApci13    terrestrial     ICRS <-> CIRS
**       iauApco iauApco13    terrestrial     ICRS <-> observed
**       iauApcs iauApcs13    space           ICRS <-> GCRS
**       iauAper iauAper13    terrestrial     update Earth rotation
**       iauApio iauApio13    terrestrial     CIRS <-> observed
**
```

```
**      Those with names ending in "13" use contemporary SOFA models to
**      compute the various ephemerides.  The others accept ephemerides
**      supplied by the caller.
**
**      The transformation from ICRS to GCRS covers space motion,
**      parallax, light deflection, and aberration.  From GCRS to CIRS
**      comprises frame bias and precession-nutation.  From CIRS to
**      observed takes account of Earth rotation, polar motion, diurnal
**      aberration and parallax (unless subsumed into the ICRS <-> GCRS
**      transformation), and atmospheric refraction.
**
**  11) The context structure astrom produced by this function is used
**      by iauAtioq and iauAtoiq.
**
**  Called:
**     iauUtctai    UTC to TAI
**     iauTaitt     TAI to TT
**     iauUtcut1    UTC to UT1
**     iauSp00      the TIO locator s', IERS 2000
**     iauEra00     Earth rotation angle, IAU 2000
**     iauRefco     refraction constants for given ambient conditions
**     iauApio      astrometry parameters, CIRS-observed
**
*/
```

```
void iauAtcc13(double rc, double dc,
               double pr, double pd, double px, double rv,
               double date1, double date2,
               double *ra, double *da)
/*
**  - - - - - - - - - -
**   i a u A t c c 1 3
**  - - - - - - - - - -
**
**  Transform a star's ICRS catalog entry (epoch J2000.0) into ICRS
**  astrometric place.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     rc     double   ICRS right ascension at J2000.0 (radians, Note 1)
**     dc     double   ICRS declination at J2000.0 (radians, Note 1)
**     pr     double   RA proper motion (radians/year, Note 2)
**     pd     double   Dec proper motion (radians/year)
**     px     double   parallax (arcsec)
**     rv     double   radial velocity (km/s, +ve if receding)
**     date1  double   TDB as a 2-part...
**     date2  double   ...Julian Date (Note 3)
**
**  Returned:
**     ra,da  double*  ICRS astrometric RA,Dec (radians)
**
**  Notes:
**
**  1) Star data for an epoch other than J2000.0 (for example from the
**     Hipparcos catalog, which has an epoch of J1991.25) will require a
**     preliminary call to iauPmsafe before use.
**
**  2) The proper motion in RA is dRA/dt rather than cos(Dec)*dRA/dt.
**
**  3) The TDB date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TDB)=2450123.7 could be expressed in any of these ways, among
**     others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in cases
**     where the loss of several decimal digits of resolution is
**     acceptable.  The J2000 method is best matched to the way the
**     argument is handled internally and will deliver the optimum
**     resolution.  The MJD method and the date & time methods are both
**     good compromises between resolution and convenience.  For most
**     applications of this function the choice will not be at all
**     critical.
**
**     TT can be used instead of TDB without any significant impact on
**     accuracy.
**
**  Called:
**     iauApci13    astrometry parameters, ICRS-CIRS, 2013
**     iauAtccq     quick catalog ICRS to astrometric
**
*/
```

```
void iauAtccq(double rc, double dc,
              double pr, double pd, double px, double rv,
              iauASTROM *astrom, double *ra, double *da)
/*
**  - - - - - - - - -
**   i a u A t c c q
**  - - - - - - - - -
**
**  Quick transformation of a star's ICRS catalog entry (epoch J2000.0)
**  into ICRS astrometric place, given precomputed star-independent
**  astrometry parameters.
**
**  Use of this function is appropriate when efficiency is important and
**  where many star positions are to be transformed for one date.  The
**  star-independent parameters can be obtained by calling one of the
**  functions iauApci[13], iauApcg[13], iauApco[13] or iauApcs[13].
**
**  If the parallax and proper motions are zero the transformation has
**  no effect.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     rc,dc  double     ICRS RA,Dec at J2000.0 (radians)
**     pr     double     RA proper motion (radians/year, Note 3)
**     pd     double     Dec proper motion (radians/year)
**     px     double     parallax (arcsec)
**     rv     double     radial velocity (km/s, +ve if receding)
**     astrom iauASTROM* star-independent astrometry parameters:
**      pmt    double       PM time interval (SSB, Julian years)
**      eb     double[3]    SSB to observer (vector, au)
**      eh     double[3]    Sun to observer (unit vector)
**      em     double       distance from Sun to observer (au)
**      v      double[3]    barycentric observer velocity (vector, c)
**      bm1    double       sqrt(1-|v|^2): reciprocal of Lorenz factor
**      bpn    double[3][3] bias-precession-nutation matrix
**      along  double       longitude + s' (radians)
**      xpl    double       polar motion xp wrt local meridian (radians)
**      ypl    double       polar motion yp wrt local meridian (radians)
**      sphi   double       sine of geodetic latitude
**      cphi   double       cosine of geodetic latitude
**      diurab double       magnitude of diurnal aberration vector
**      eral   double       "local" Earth rotation angle (radians)
**      refa   double       refraction constant A (radians)
**      refb   double       refraction constant B (radians)
**
**  Returned:
**     ra,da  double*    ICRS astrometric RA,Dec (radians)
**
**  Notes:
**
**  1) All the vectors are with respect to BCRS axes.
**
**  2) Star data for an epoch other than J2000.0 (for example from the
**     Hipparcos catalog, which has an epoch of J1991.25) will require a
**     preliminary call to iauPmsafe before use.
**
**  3) The proper motion in RA is dRA/dt rather than cos(Dec)*dRA/dt.
**
**  Called:
**     iauPmpx      proper motion and parallax
**     iauC2s       p-vector to spherical
**     iauAnp       normalize angle into range 0 to 2pi
**
*/
```

```
void iauAtci13(double rc, double dc,
               double pr, double pd, double px, double rv,
               double date1, double date2,
               double *ri, double *di, double *eo)
/*
**  - - - - - - - - - -
**   i a u A t c i 1 3
**  - - - - - - - - - -
**
**  Transform ICRS star data, epoch J2000.0, to CIRS.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     rc     double  ICRS right ascension at J2000.0 (radians, Note 1)
**     dc     double  ICRS declination at J2000.0 (radians, Note 1)
**     pr     double  RA proper motion (radians/year, Note 2)
**     pd     double  Dec proper motion (radians/year)
**     px     double  parallax (arcsec)
**     rv     double  radial velocity (km/s, +ve if receding)
**     date1  double  TDB as a 2-part...
**     date2  double  ...Julian Date (Note 3)
**
**  Returned:
**     ri,di  double* CIRS geocentric RA,Dec (radians)
**     eo     double* equation of the origins (ERA-GST, radians, Note 5)
**
**  Notes:
**
**  1) Star data for an epoch other than J2000.0 (for example from the
**     Hipparcos catalog, which has an epoch of J1991.25) will require a
**     preliminary call to iauPmsafe before use.
**
**  2) The proper motion in RA is dRA/dt rather than cos(Dec)*dRA/dt.
**
**  3) The TDB date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TDB)=2450123.7 could be expressed in any of these ways, among
**     others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in cases
**     where the loss of several decimal digits of resolution is
**     acceptable.  The J2000 method is best matched to the way the
**     argument is handled internally and will deliver the optimum
**     resolution.  The MJD method and the date & time methods are both
**     good compromises between resolution and convenience.  For most
**     applications of this function the choice will not be at all
**     critical.
**
**     TT can be used instead of TDB without any significant impact on
**     accuracy.
**
**  4) The available accuracy is better than 1 milliarcsecond, limited
**     mainly by the precession-nutation model that is used, namely
**     IAU 2000A/2006.  Very close to solar system bodies, additional
**     errors of up to several milliarcseconds can occur because of
**     unmodeled light deflection;  however, the Sun's contribution is
**     taken into account, to first order.  The accuracy limitations of
**     the SOFA function iauEpv00 (used to compute Earth position and
**     velocity) can contribute aberration errors of up to
**     5 microarcseconds.  Light deflection at the Sun's limb is
```

```
**     uncertain at the 0.4 mas level.
**
**  5) Should the transformation to (equinox based) apparent place be
**     required rather than (CIO based) intermediate place, subtract the
**     equation of the origins from the returned right ascension:
**     RA = RI - EO. (The iauAnp function can then be applied, as
**     required, to keep the result in the conventional 0-2pi range.)
**
**  Called:
**     iauApci13    astrometry parameters, ICRS-CIRS, 2013
**     iauAtciq     quick ICRS to CIRS
**
*/
```

```
void iauAtciq(double rc, double dc,
              double pr, double pd, double px, double rv,
              iauASTROM *astrom, double *ri, double *di)
/*
**  - - - - - - - - -
**   i a u A t c i q
**  - - - - - - - - -
**
**  Quick ICRS, epoch J2000.0, to CIRS transformation, given precomputed
**  star-independent astrometry parameters.
**
**  Use of this function is appropriate when efficiency is important and
**  where many star positions are to be transformed for one date.  The
**  star-independent parameters can be obtained by calling one of the
**  functions iauApci[13], iauApcg[13], iauApco[13] or iauApcs[13].
**
**  If the parallax and proper motions are zero the iauAtciqz function
**  can be used instead.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     rc,dc  double     ICRS RA,Dec at J2000.0 (radians, Note 1)
**     pr     double     RA proper motion (radians/year, Note 2)
**     pd     double     Dec proper motion (radians/year)
**     px     double     parallax (arcsec)
**     rv     double     radial velocity (km/s, +ve if receding)
**     astrom iauASTROM* star-independent astrometry parameters:
**      pmt    double       PM time interval (SSB, Julian years)
**      eb     double[3]    SSB to observer (vector, au)
**      eh     double[3]    Sun to observer (unit vector)
**      em     double       distance from Sun to observer (au)
**      v      double[3]    barycentric observer velocity (vector, c)
**      bm1    double       sqrt(1-|v|^2): reciprocal of Lorenz factor
**      bpn    double[3][3] bias-precession-nutation matrix
**      along  double       longitude + s' (radians)
**      xpl    double       polar motion xp wrt local meridian (radians)
**      ypl    double       polar motion yp wrt local meridian (radians)
**      sphi   double       sine of geodetic latitude
**      cphi   double       cosine of geodetic latitude
**      diurab double       magnitude of diurnal aberration vector
**      eral   double       "local" Earth rotation angle (radians)
**      refa   double       refraction constant A (radians)
**      refb   double       refraction constant B (radians)
**
**  Returned:
**     ri,di   double    CIRS RA,Dec (radians)
**
**  Notes:
**
**  1) Star data for an epoch other than J2000.0 (for example from the
**     Hipparcos catalog, which has an epoch of J1991.25) will require a
**     preliminary call to iauPmsafe before use.
**
**  2) The proper motion in RA is dRA/dt rather than cos(Dec)*dRA/dt.
**
**  Called:
**     iauPmpx      proper motion and parallax
**     iauLdsun     light deflection by the Sun
**     iauAb        stellar aberration
**     iauRxp       product of r-matrix and pv-vector
**     iauC2s       p-vector to spherical
**     iauAnp       normalize angle into range 0 to 2pi
**
*/
```

```
void iauAtciqn(double rc, double dc, double pr, double pd,
               double px, double rv, iauASTROM *astrom,
               int n, iauLDBODY b[], double *ri, double *di)
/*
**  - - - - - - - - - -
**   i a u A t c i q n
**  - - - - - - - - - -
**
**  Quick ICRS, epoch J2000.0, to CIRS transformation, given precomputed
**  star-independent astrometry parameters plus a list of light-
**  deflecting bodies.
**
**  Use of this function is appropriate when efficiency is important and
**  where many star positions are to be transformed for one date.  The
**  star-independent parameters can be obtained by calling one of the
**  functions iauApci[13], iauApcg[13], iauApco[13] or iauApcs[13].
**
**
**  If the only light-deflecting body to be taken into account is the
**  Sun, the iauAtciq function can be used instead.  If in addition the
**  parallax and proper motions are zero, the iauAtciqz function can be
**  used.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     rc,dc  double       ICRS RA,Dec at J2000.0 (radians)
**     pr     double       RA proper motion (radians/year, Note 3)
**     pd     double       Dec proper motion (radians/year)
**     px     double       parallax (arcsec)
**     rv     double       radial velocity (km/s, +ve if receding)
**     astrom iauASTROM*   star-independent astrometry parameters:
**      pmt    double       PM time interval (SSB, Julian years)
**      eb     double[3]    SSB to observer (vector, au)
**      eh     double[3]    Sun to observer (unit vector)
**      em     double       distance from Sun to observer (au)
**      v      double[3]    barycentric observer velocity (vector, c)
**      bm1    double       sqrt(1-|v|^2): reciprocal of Lorenz factor
**      bpn    double[3][3] bias-precession-nutation matrix
**      along  double       longitude + s' (radians)
**      xpl    double       polar motion xp wrt local meridian (radians)
**      ypl    double       polar motion yp wrt local meridian (radians)
**      sphi   double       sine of geodetic latitude
**      cphi   double       cosine of geodetic latitude
**      diurab double       magnitude of diurnal aberration vector
**      eral   double       "local" Earth rotation angle (radians)
**      refa   double       refraction constant A (radians)
**      refb   double       refraction constant B (radians)
**     n      int          number of bodies (Note 3)
**     b      iauLDBODY[n] data for each of the n bodies (Notes 3,4):
**      bm     double       mass of the body (solar masses, Note 5)
**      dl     double       deflection limiter (Note 6)
**      pv     [2][3]       barycentric PV of the body (au, au/day)
**
**  Returned:
**     ri,di   double    CIRS RA,Dec (radians)
**
**  Notes:
**
**  1) Star data for an epoch other than J2000.0 (for example from the
**     Hipparcos catalog, which has an epoch of J1991.25) will require a
**     preliminary call to iauPmsafe before use.
**
**  2) The proper motion in RA is dRA/dt rather than cos(Dec)*dRA/dt.
**
**  3) The struct b contains n entries, one for each body to be
**     considered.  If n = 0, no gravitational light deflection will be
**     applied, not even for the Sun.
```

```
**
**  4) The struct b should include an entry for the Sun as well as for
**     any planet or other body to be taken into account.  The entries
**     should be in the order in which the light passes the body.
**
**  5) In the entry in the b struct for body i, the mass parameter
**     b[i].bm can, as required, be adjusted in order to allow for such
**     effects as quadrupole field.
**
**  6) The deflection limiter parameter b[i].dl is phi^2/2, where phi is
**     the angular separation (in radians) between star and body at
**     which limiting is applied.  As phi shrinks below the chosen
**     threshold, the deflection is artificially reduced, reaching zero
**     for phi = 0.   Example values suitable for a terrestrial
**     observer, together with masses, are as follows:
**
**        body i     b[i].bm        b[i].dl
**
**        Sun        1.0            6e-6
**        Jupiter    0.00095435     3e-9
**        Saturn     0.00028574     3e-10
**
**  7) For efficiency, validation of the contents of the b array is
**     omitted.  The supplied masses must be greater than zero, the
**     position and velocity vectors must be right, and the deflection
**     limiter greater than zero.
**
**  Called:
**     iauPmpx      proper motion and parallax
**     iauLdn       light deflection by n bodies
**     iauAb        stellar aberration
**     iauRxp       product of r-matrix and pv-vector
**     iauC2s       p-vector to spherical
**     iauAnp       normalize angle into range 0 to 2pi
**
*/
```

```
void iauAtciqz(double rc, double dc, iauASTROM *astrom,
               double *ri, double *di)
/*
**  - - - - - - - - - -
**   i a u A t c i q z
**  - - - - - - - - - -
**
**  Quick ICRS to CIRS transformation, given precomputed star-
**  independent astrometry parameters, and assuming zero parallax and
**  proper motion.
**
**  Use of this function is appropriate when efficiency is important and
**  where many star positions are to be transformed for one date.  The
**  star-independent parameters can be obtained by calling one of the
**  functions iauApci[13], iauApcg[13], iauApco[13] or iauApcs[13].
**
**  The corresponding function for the case of non-zero parallax and
**  proper motion is iauAtciq.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     rc,dc  double     ICRS astrometric RA,Dec (radians)
**     astrom iauASTROM* star-independent astrometry parameters:
**      pmt    double       PM time interval (SSB, Julian years)
**      eb     double[3]    SSB to observer (vector, au)
**      eh     double[3]    Sun to observer (unit vector)
**      em     double       distance from Sun to observer (au)
**      v      double[3]    barycentric observer velocity (vector, c)
**      bm1    double       sqrt(1-|v|^2): reciprocal of Lorenz factor
**      bpn    double[3][3] bias-precession-nutation matrix
**      along  double       longitude + s' (radians)
**      xpl    double       polar motion xp wrt local meridian (radians)
**      ypl    double       polar motion yp wrt local meridian (radians)
**      sphi   double       sine of geodetic latitude
**      cphi   double       cosine of geodetic latitude
**      diurab double       magnitude of diurnal aberration vector
**      eral   double       "local" Earth rotation angle (radians)
**      refa   double       refraction constant A (radians)
**      refb   double       refraction constant B (radians)
**
**  Returned:
**     ri,di  double     CIRS RA,Dec (radians)
**
**  Note:
**
**     All the vectors are with respect to BCRS axes.
**
**  References:
**
**     Urban, S. & Seidelmann, P. K. (eds), Explanatory Supplement to
**     the Astronomical Almanac, 3rd ed., University Science Books
**     (2013).
**
**     Klioner, Sergei A., "A practical relativistic model for micro-
**     arcsecond astrometry in space", Astr. J. 125, 1580-1597 (2003).
**
**  Called:
**     iauS2c       spherical coordinates to unit vector
**     iauLdsun     light deflection due to Sun
**     iauAb        stellar aberration
**     iauRxp       product of r-matrix and p-vector
**     iauC2s       p-vector to spherical
**     iauAnp       normalize angle into range +/- pi
**
*/
```

```
int iauAtco13(double rc, double dc,
              double pr, double pd, double px, double rv,
              double utc1, double utc2, double dut1,
              double elong, double phi, double hm, double xp, double yp,
              double phpa, double tc, double rh, double wl,
              double *aob, double *zob, double *hob,
              double *dob, double *rob, double *eo)
/*
**  - - - - - - - - - -
**   i a u A t c o 1 3
**  - - - - - - - - - -
**
**  ICRS RA,Dec to observed place.  The caller supplies UTC, site
**  coordinates, ambient air conditions and observing wavelength.
**
**  SOFA models are used for the Earth ephemeris, bias-precession-
**  nutation, Earth orientation and refraction.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     rc,dc  double   ICRS right ascension at J2000.0 (radians, Note 1)
**     pr     double   RA proper motion (radians/year, Note 2)
**     pd     double   Dec proper motion (radians/year)
**     px     double   parallax (arcsec)
**     rv     double   radial velocity (km/s, +ve if receding)
**     utc1   double   UTC as a 2-part...
**     utc2   double   ...quasi Julian Date (Notes 3-4)
**     dut1   double   UT1-UTC (seconds, Note 5)
**     elong  double   longitude (radians, east +ve, Note 6)
**     phi    double   latitude (geodetic, radians, Note 6)
**     hm     double   height above ellipsoid (m, geodetic, Notes 6,8)
**     xp,yp  double   polar motion coordinates (radians, Note 7)
**     phpa   double   pressure at the observer (hPa = mB, Note 8)
**     tc     double   ambient temperature at the observer (deg C)
**     rh     double   relative humidity at the observer (range 0-1)
**     wl     double   wavelength (micrometers, Note 9)
**
**  Returned:
**     aob    double*  observed azimuth (radians: N=0,E=90)
**     zob    double*  observed zenith distance (radians)
**     hob    double*  observed hour angle (radians)
**     dob    double*  observed declination (radians)
**     rob    double*  observed right ascension (CIO-based, radians)
**     eo     double*  equation of the origins (ERA-GST, radians)
**
**  Returned (function value):
**            int      status: +1 = dubious year (Note 4)
**                              0 = OK
**                             -1 = unacceptable date
**
**  Notes:
**
**  1)  Star data for an epoch other than J2000.0 (for example from the
**      Hipparcos catalog, which has an epoch of J1991.25) will require
**      a preliminary call to iauPmsafe before use.
**
**  2)  The proper motion in RA is dRA/dt rather than cos(Dec)*dRA/dt.
**
**  3)  utc1+utc2 is quasi Julian Date (see Note 2), apportioned in any
**      convenient way between the two arguments, for example where utc1
**      is the Julian Day Number and utc2 is the fraction of a day.
**
**      However, JD cannot unambiguously represent UTC during a leap
**      second unless special measures are taken.  The convention in the
**      present function is that the JD day represents UTC days whether
**      the length is 86399, 86400 or 86401 SI seconds.
**
```

```
**      Applications should use the function iauDtf2d to convert from
**      calendar date and time of day into 2-part quasi Julian Date, as
**      it implements the leap-second-ambiguity convention just
**      described.
**
**   4)  The warning status "dubious year" flags UTCs that predate the
**      introduction of the time scale or that are too far in the
**      future to be trusted.  See iauDat for further details.
**
**   5)  UT1-UTC is tabulated in IERS bulletins.  It increases by exactly
**      one second at the end of each positive UTC leap second,
**      introduced in order to keep UT1-UTC within +/- 0.9s.  n.b. This
**      practice is under review, and in the future UT1-UTC may grow
**      essentially without limit.
**
**   6)  The geographical coordinates are with respect to the WGS84
**      reference ellipsoid.  TAKE CARE WITH THE LONGITUDE SIGN:  the
**      longitude required by the present function is east-positive
**      (i.e. right-handed), in accordance with geographical convention.
**
**   7)  The polar motion xp,yp can be obtained from IERS bulletins.  The
**      values are the coordinates (in radians) of the Celestial
**      Intermediate Pole with respect to the International Terrestrial
**      Reference System (see IERS Conventions 2003), measured along the
**      meridians 0 and 90 deg west respectively.  For many
**      applications, xp and yp can be set to zero.
**
**   8)  If hm, the height above the ellipsoid of the observing station
**      in meters, is not known but phpa, the pressure in hPa (=mB),
**      is available, an adequate estimate of hm can be obtained from
**      the expression
**
**            hm = -29.3 * tsl * log ( phpa / 1013.25 );
**
**      where tsl is the approximate sea-level air temperature in K
**      (See Astrophysical Quantities, C.W.Allen, 3rd edition, section
**      52).  Similarly, if the pressure phpa is not known, it can be
**      estimated from the height of the observing station, hm, as
**      follows:
**
**            phpa = 1013.25 * exp ( -hm / ( 29.3 * tsl ) );
**
**      Note, however, that the refraction is nearly proportional to
**      the pressure and that an accurate phpa value is important for
**      precise work.
**
**   9)  The argument wl specifies the observing wavelength in
**      micrometers.  The transition from optical to radio is assumed to
**      occur at 100 micrometers (about 3000 GHz).
**
**   10) The accuracy of the result is limited by the corrections for
**      refraction, which use a simple A*tan(z) + B*tan^3(z) model.
**      Providing the meteorological parameters are known accurately and
**      there are no gross local effects, the predicted observed
**      coordinates should be within 0.05 arcsec (optical) or 1 arcsec
**      (radio) for a zenith distance of less than 70 degrees, better
**      than 30 arcsec (optical or radio) at 85 degrees and better
**      than 20 arcmin (optical) or 30 arcmin (radio) at the horizon.
**
**      Without refraction, the complementary functions iauAtco13 and
**      iauAtoc13 are self-consistent to better than 1 microarcsecond
**      all over the celestial sphere.  With refraction included,
**      consistency falls off at high zenith distances, but is still
**      better than 0.05 arcsec at 85 degrees.
**
**   11) "Observed" Az,ZD means the position that would be seen by a
**      perfect geodetically aligned theodolite.  (Zenith distance is
**      used rather than altitude in order to reflect the fact that no
**      allowance is made for depression of the horizon.)  This is
**      related to the observed HA,Dec via the standard rotation, using
**      the geodetic latitude (corrected for polar motion), while the
**      observed HA and RA are related simply through the Earth rotation
**      angle and the site longitude.  "Observed" RA,Dec or HA,Dec thus
```

```
**      means the position that would be seen by a perfect equatorial
**      with its polar axis aligned to the Earth's axis of rotation.
**
**  12) It is advisable to take great care with units, as even unlikely
**      values of the input parameters are accepted and processed in
**      accordance with the models used.
**
**  Called:
**     iauApco13    astrometry parameters, ICRS-observed, 2013
**     iauAtciq     quick ICRS to CIRS
**     iauAtioq     quick CIRS to observed
**
*/
```

```
void iauAtic13(double ri, double di, double date1, double date2,
               double *rc, double *dc, double *eo)
/*
**  - - - - - - - - - -
**   i a u A t i c 1 3
**  - - - - - - - - - -
**
**  Transform star RA,Dec from geocentric CIRS to ICRS astrometric.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     ri,di  double  CIRS geocentric RA,Dec (radians)
**     date1  double  TDB as a 2-part...
**     date2  double  ...Julian Date (Note 1)
**
**  Returned:
**     rc,dc  double  ICRS astrometric RA,Dec (radians)
**     eo     double  equation of the origins (ERA-GST, radians, Note 4)
**
**  Notes:
**
**  1) The TDB date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TDB)=2450123.7 could be expressed in any of these ways, among
**     others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in cases
**     where the loss of several decimal digits of resolution is
**     acceptable.  The J2000 method is best matched to the way the
**     argument is handled internally and will deliver the optimum
**     resolution.  The MJD method and the date & time methods are both
**     good compromises between resolution and convenience.  For most
**     applications of this function the choice will not be at all
**     critical.
**
**     TT can be used instead of TDB without any significant impact on
**     accuracy.
**
**  2) Iterative techniques are used for the aberration and light
**     deflection corrections so that the functions iauAtic13 (or
**     iauAticq) and iauAtci13 (or iauAtciq) are accurate inverses;
**     even at the edge of the Sun's disk the discrepancy is only about
**     1 nanoarcsecond.
**
**  3) The available accuracy is better than 1 milliarcsecond, limited
**     mainly by the precession-nutation model that is used, namely
**     IAU 2000A/2006.  Very close to solar system bodies, additional
**     errors of up to several milliarcseconds can occur because of
**     unmodeled light deflection;  however, the Sun's contribution is
**     taken into account, to first order.  The accuracy limitations of
**     the SOFA function iauEpv00 (used to compute Earth position and
**     velocity) can contribute aberration errors of up to
**     5 microarcseconds.  Light deflection at the Sun's limb is
**     uncertain at the 0.4 mas level.
**
**  4) Should the transformation to (equinox based) J2000.0 mean place
**     be required rather than (CIO based) ICRS coordinates, subtract the
**     equation of the origins from the returned right ascension:
**     RA = RI - EO.  (The iauAnp function can then be applied, as
**     required, to keep the result in the conventional 0-2pi range.)
```

```
**
**  Called:
**     iauApci13    astrometry parameters, ICRS-CIRS, 2013
**     iauAticq     quick CIRS to ICRS astrometric
**
*/
```

```
void iauAticq(double ri, double di, iauASTROM *astrom,
              double *rc, double *dc)
/*
**  - - - - - - - - -
**   i a u A t i c q
**  - - - - - - - - -
**
**  Quick CIRS RA,Dec to ICRS astrometric place, given the star-
**  independent astrometry parameters.
**
**  Use of this function is appropriate when efficiency is important and
**  where many star positions are all to be transformed for one date.
**  The star-independent astrometry parameters can be obtained by
**  calling one of the functions iauApci[13], iauApcg[13], iauApco[13]
**  or iauApcs[13].
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     ri,di  double     CIRS RA,Dec (radians)
**     astrom iauASTROM* star-independent astrometry parameters:
**      pmt    double       PM time interval (SSB, Julian years)
**      eb     double[3]    SSB to observer (vector, au)
**      eh     double[3]    Sun to observer (unit vector)
**      em     double       distance from Sun to observer (au)
**      v      double[3]    barycentric observer velocity (vector, c)
**      bm1    double       sqrt(1-|v|^2): reciprocal of Lorenz factor
**      bpn    double[3][3] bias-precession-nutation matrix
**      along  double       longitude + s' (radians)
**      xpl    double       polar motion xp wrt local meridian (radians)
**      ypl    double       polar motion yp wrt local meridian (radians)
**      sphi   double       sine of geodetic latitude
**      cphi   double       cosine of geodetic latitude
**      diurab double       magnitude of diurnal aberration vector
**      eral   double       "local" Earth rotation angle (radians)
**      refa   double       refraction constant A (radians)
**      refb   double       refraction constant B (radians)
**
**  Returned:
**     rc,dc  double     ICRS astrometric RA,Dec (radians)
**
**  Notes:
**
**  1) Only the Sun is taken into account in the light deflection
**     correction.
**
**  2) Iterative techniques are used for the aberration and light
**     deflection corrections so that the functions iauAtic13 (or
**     iauAticq) and iauAtci13 (or iauAtciq) are accurate inverses;
**     even at the edge of the Sun's disk the discrepancy is only about
**     1 nanoarcsecond.
**
**  Called:
**     iauS2c       spherical coordinates to unit vector
**     iauTrxp      product of transpose of r-matrix and p-vector
**     iauZp        zero p-vector
**     iauAb        stellar aberration
**     iauLdsun     light deflection by the Sun
**     iauC2s       p-vector to spherical
**     iauAnp       normalize angle into range +/- pi
**
*/
```

```
void iauAticqn(double ri, double di, iauASTROM *astrom,
               int n, iauLDBODY b[], double *rc, double *dc)
/*
**  - - - - - - - - - - -
**   i a u A t i c q n
**  - - - - - - - - - - -
**
**  Quick CIRS to ICRS astrometric place transformation, given the star-
**  independent astrometry parameters plus a list of light-deflecting
**  bodies.
**
**  Use of this function is appropriate when efficiency is important and
**  where many star positions are all to be transformed for one date.
**  The star-independent astrometry parameters can be obtained by
**  calling one of the functions iauApci[13], iauApcg[13], iauApco[13]
**  or iauApcs[13].
*
*  If the only light-deflecting body to be taken into account is the
*  Sun, the iauAticq function can be used instead.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     ri,di  double      CIRS RA,Dec (radians)
**     astrom iauASTROM*  star-independent astrometry parameters:
**      pmt    double       PM time interval (SSB, Julian years)
**      eb     double[3]    SSB to observer (vector, au)
**      eh     double[3]    Sun to observer (unit vector)
**      em     double       distance from Sun to observer (au)
**      v      double[3]    barycentric observer velocity (vector, c)
**      bm1    double       sqrt(1-|v|^2): reciprocal of Lorenz factor
**      bpn    double[3][3] bias-precession-nutation matrix
**      along  double       longitude + s' (radians)
**      xpl    double       polar motion xp wrt local meridian (radians)
**      ypl    double       polar motion yp wrt local meridian (radians)
**      sphi   double       sine of geodetic latitude
**      cphi   double       cosine of geodetic latitude
**      diurab double       magnitude of diurnal aberration vector
**      eral   double       "local" Earth rotation angle (radians)
**      refa   double       refraction constant A (radians)
**      refb   double       refraction constant B (radians)
**     n      int          number of bodies (Note 3)
**     b      iauLDBODY[n] data for each of the n bodies (Notes 3,4):
**      bm     double       mass of the body (solar masses, Note 5)
**      dl     double       deflection limiter (Note 6)
**      pv     [2][3]       barycentric PV of the body (au, au/day)
**
**  Returned:
**     rc,dc  double     ICRS astrometric RA,Dec (radians)
**
**  Notes:
**
**  1) Iterative techniques are used for the aberration and light
**     deflection corrections so that the functions iauAticqn and
**     iauAtciqn are accurate inverses; even at the edge of the Sun's
**     disk the discrepancy is only about 1 nanoarcsecond.
**
**  2) If the only light-deflecting body to be taken into account is the
**     Sun, the iauAticq function can be used instead.
**
**  3) The struct b contains n entries, one for each body to be
**     considered.  If n = 0, no gravitational light deflection will be
**     applied, not even for the Sun.
**
**  4) The struct b should include an entry for the Sun as well as for
**     any planet or other body to be taken into account.  The entries
**     should be in the order in which the light passes the body.
**
```

```
**  5) In the entry in the b struct for body i, the mass parameter
**     b[i].bm can, as required, be adjusted in order to allow for such
**     effects as quadrupole field.
**
**  6) The deflection limiter parameter b[i].dl is phi^2/2, where phi is
**     the angular separation (in radians) between star and body at
**     which limiting is applied.  As phi shrinks below the chosen
**     threshold, the deflection is artificially reduced, reaching zero
**     for phi = 0.   Example values suitable for a terrestrial
**     observer, together with masses, are as follows:
**
**        body i     b[i].bm        b[i].dl
**
**        Sun        1.0            6e-6
**        Jupiter    0.00095435     3e-9
**        Saturn     0.00028574     3e-10
**
**  7) For efficiency, validation of the contents of the b array is
**     omitted.  The supplied masses must be greater than zero, the
**     position and velocity vectors must be right, and the deflection
**     limiter greater than zero.
**
**  Called:
**     iauS2c       spherical coordinates to unit vector
**     iauTrxp      product of transpose of r-matrix and p-vector
**     iauZp        zero p-vector
**     iauAb        stellar aberration
**     iauLdn       light deflection by n bodies
**     iauC2s       p-vector to spherical
**     iauAnp       normalize angle into range +/- pi
**
*/
```

```
int iauAtio13(double ri, double di,
              double utc1, double utc2, double dut1,
              double elong, double phi, double hm, double xp, double yp,
              double phpa, double tc, double rh, double wl,
              double *aob, double *zob, double *hob,
              double *dob, double *rob)
/*
**  - - - - - - - - - - 
**   i a u A t i o 1 3
**  - - - - - - - - - - 
**
**  CIRS RA,Dec to observed place.  The caller supplies UTC, site
**  coordinates, ambient air conditions and observing wavelength.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     ri     double   CIRS right ascension (CIO-based, radians)
**     di     double   CIRS declination (radians)
**     utc1   double   UTC as a 2-part...
**     utc2   double   ...quasi Julian Date (Notes 1,2)
**     dut1   double   UT1-UTC (seconds, Note 3)
**     elong  double   longitude (radians, east +ve, Note 4)
**     phi    double   geodetic latitude (radians, Note 4)
**     hm     double   height above ellipsoid (m, geodetic Notes 4,6)
**     xp,yp  double   polar motion coordinates (radians, Note 5)
**     phpa   double   pressure at the observer (hPa = mB, Note 6)
**     tc     double   ambient temperature at the observer (deg C)
**     rh     double   relative humidity at the observer (range 0-1)
**     wl     double   wavelength (micrometers, Note 7)
**
**  Returned:
**     aob    double*  observed azimuth (radians: N=0,E=90)
**     zob    double*  observed zenith distance (radians)
**     hob    double*  observed hour angle (radians)
**     dob    double*  observed declination (radians)
**     rob    double*  observed right ascension (CIO-based, radians)
**
**  Returned (function value):
**            int      status: +1 = dubious year (Note 2)
**                              0 = OK
**                             -1 = unacceptable date
**
**  Notes:
**
**  1)  utc1+utc2 is quasi Julian Date (see Note 2), apportioned in any
**      convenient way between the two arguments, for example where utc1
**      is the Julian Day Number and utc2 is the fraction of a day.
**
**      However, JD cannot unambiguously represent UTC during a leap
**      second unless special measures are taken.  The convention in the
**      present function is that the JD day represents UTC days whether
**      the length is 86399, 86400 or 86401 SI seconds.
**
**      Applications should use the function iauDtf2d to convert from
**      calendar date and time of day into 2-part quasi Julian Date, as
**      it implements the leap-second-ambiguity convention just
**      described.
**
**  2)  The warning status "dubious year" flags UTCs that predate the
**      introduction of the time scale or that are too far in the
**      future to be trusted.  See iauDat for further details.
**
**  3)  UT1-UTC is tabulated in IERS bulletins.  It increases by exactly
**      one second at the end of each positive UTC leap second,
**      introduced in order to keep UT1-UTC within +/- 0.9s.  n.b. This
**      practice is under review, and in the future UT1-UTC may grow
**      essentially without limit.
```

```
**
**  4)  The geographical coordinates are with respect to the WGS84
**      reference ellipsoid.  TAKE CARE WITH THE LONGITUDE SIGN:  the
**      longitude required by the present function is east-positive
**      (i.e. right-handed), in accordance with geographical convention.
**
**  5)  The polar motion xp,yp can be obtained from IERS bulletins.  The
**      values are the coordinates (in radians) of the Celestial
**      Intermediate Pole with respect to the International Terrestrial
**      Reference System (see IERS Conventions 2003), measured along the
**      meridians 0 and 90 deg west respectively.  For many
**      applications, xp and yp can be set to zero.
**
**  6)  If hm, the height above the ellipsoid of the observing station
**      in meters, is not known but phpa, the pressure in hPa (=mB), is
**      available, an adequate estimate of hm can be obtained from the
**      expression
**
**            hm = -29.3 * tsl * log ( phpa / 1013.25 );
**
**      where tsl is the approximate sea-level air temperature in K
**      (See Astrophysical Quantities, C.W.Allen, 3rd edition, section
**      52).  Similarly, if the pressure phpa is not known, it can be
**      estimated from the height of the observing station, hm, as
**      follows:
**
**            phpa = 1013.25 * exp ( -hm / ( 29.3 * tsl ) );
**
**      Note, however, that the refraction is nearly proportional to
**      the pressure and that an accurate phpa value is important for
**      precise work.
**
**  7)  The argument wl specifies the observing wavelength in
**      micrometers.  The transition from optical to radio is assumed to
**      occur at 100 micrometers (about 3000 GHz).
**
**  8)  "Observed" Az,ZD means the position that would be seen by a
**      perfect geodetically aligned theodolite.  (Zenith distance is
**      used rather than altitude in order to reflect the fact that no
**      allowance is made for depression of the horizon.)  This is
**      related to the observed HA,Dec via the standard rotation, using
**      the geodetic latitude (corrected for polar motion), while the
**      observed HA and RA are related simply through the Earth rotation
**      angle and the site longitude.  "Observed" RA,Dec or HA,Dec thus
**      means the position that would be seen by a perfect equatorial
**      with its polar axis aligned to the Earth's axis of rotation.
**
**  9)  The accuracy of the result is limited by the corrections for
**      refraction, which use a simple A*tan(z) + B*tan^3(z) model.
**      Providing the meteorological parameters are known accurately and
**      there are no gross local effects, the predicted astrometric
**      coordinates should be within 0.05 arcsec (optical) or 1 arcsec
**      (radio) for a zenith distance of less than 70 degrees, better
**      than 30 arcsec (optical or radio) at 85 degrees and better
**      than 20 arcmin (optical) or 30 arcmin (radio) at the horizon.
**
**  10) The complementary functions iauAtio13 and iauAtoi13 are self-
**      consistent to better than 1 microarcsecond all over the
**      celestial sphere.
**
**  11) It is advisable to take great care with units, as even unlikely
**      values of the input parameters are accepted and processed in
**      accordance with the models used.
**
**  Called:
**     iauApio13    astrometry parameters, CIRS-observed, 2013
**     iauAtioq     quick CIRS to observed
**
*/
```

```
void iauAtioq(double ri, double di, iauASTROM *astrom,
              double *aob, double *zob,
              double *hob, double *dob, double *rob)
/*
**  - - - - - - - - -
**   i a u A t i o q
**  - - - - - - - - -
**
**  Quick CIRS to observed place transformation.
**
**  Use of this function is appropriate when efficiency is important and
**  where many star positions are all to be transformed for one date.
**  The star-independent astrometry parameters can be obtained by
**  calling iauApio[13] or iauApco[13].
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     ri     double     CIRS right ascension
**     di     double     CIRS declination
**     astrom iauASTROM* star-independent astrometry parameters:
**      pmt    double       PM time interval (SSB, Julian years)
**      eb     double[3]    SSB to observer (vector, au)
**      eh     double[3]    Sun to observer (unit vector)
**      em     double       distance from Sun to observer (au)
**      v      double[3]    barycentric observer velocity (vector, c)
**      bm1    double       sqrt(1-|v|^2): reciprocal of Lorenz factor
**      bpn    double[3][3] bias-precession-nutation matrix
**      along  double       longitude + s' (radians)
**      xpl    double       polar motion xp wrt local meridian (radians)
**      ypl    double       polar motion yp wrt local meridian (radians)
**      sphi   double       sine of geodetic latitude
**      cphi   double       cosine of geodetic latitude
**      diurab double       magnitude of diurnal aberration vector
**      eral   double       "local" Earth rotation angle (radians)
**      refa   double       refraction constant A (radians)
**      refb   double       refraction constant B (radians)
**
**  Returned:
**     aob    double*    observed azimuth (radians: N=0,E=90)
**     zob    double*    observed zenith distance (radians)
**     hob    double*    observed hour angle (radians)
**     dob    double*    observed declination (radians)
**     rob    double*    observed right ascension (CIO-based, radians)
**
**  Notes:
**
**  1) This function returns zenith distance rather than altitude in
**     order to reflect the fact that no allowance is made for
**     depression of the horizon.
**
**  2) The accuracy of the result is limited by the corrections for
**     refraction, which use a simple A*tan(z) + B*tan^3(z) model.
**     Providing the meteorological parameters are known accurately and
**     there are no gross local effects, the predicted observed
**     coordinates should be within 0.05 arcsec (optical) or 1 arcsec
**     (radio) for a zenith distance of less than 70 degrees, better
**     than 30 arcsec (optical or radio) at 85 degrees and better
**     than 20 arcmin (optical) or 30 arcmin (radio) at the horizon.
**
**     Without refraction, the complementary functions iauAtioq and
**     iauAtoiq are self-consistent to better than 1 microarcsecond all
**     over the celestial sphere.  With refraction included, consistency
**     falls off at high zenith distances, but is still better than
**     0.05 arcsec at 85 degrees.
**
**  3) It is advisable to take great care with units, as even unlikely
**     values of the input parameters are accepted and processed in
```

```
**     accordance with the models used.
**
**  4) The CIRS RA,Dec is obtained from a star catalog mean place by
**     allowing for space motion, parallax, the Sun's gravitational lens
**     effect, annual aberration and precession-nutation.  For star
**     positions in the ICRS, these effects can be applied by means of
**     the iauAtci13 (etc.) functions.  Starting from classical "mean
**     place" systems, additional transformations will be needed first.
**
**  5) "Observed" Az,El means the position that would be seen by a
**     perfect geodetically aligned theodolite.  This is obtained from
**     the CIRS RA,Dec by allowing for Earth orientation and diurnal
**     aberration, rotating from equator to horizon coordinates, and
**     then adjusting for refraction.  The HA,Dec is obtained by
**     rotating back into equatorial coordinates, and is the position
**     that would be seen by a perfect equatorial with its polar axis
**     aligned to the Earth's axis of rotation.  Finally, the
**     (CIO-based) RA is obtained by subtracting the HA from the local
**     ERA.
**
**  6) The star-independent CIRS-to-observed-place parameters in ASTROM
**     may be computed with iauApio[13] or iauApco[13].  If nothing has
**     changed significantly except the time, iauAper[13] may be used to
**     perform the requisite adjustment to the astrom structure.
**
**  Called:
**     iauS2c       spherical coordinates to unit vector
**     iauC2s       p-vector to spherical
**     iauAnp       normalize angle into range 0 to 2pi
**
*/
```

```
int iauAtoc13(const char *type, double ob1, double ob2,
              double utc1, double utc2, double dut1,
              double elong, double phi, double hm, double xp, double yp,
              double phpa, double tc, double rh, double wl,
              double *rc, double *dc)
/*
**  - - - - - - - - - -
**   i a u A t o c 1 3
**  - - - - - - - - - -
**
**  Observed place at a groundbased site to to ICRS astrometric RA,Dec.
**  The caller supplies UTC, site coordinates, ambient air conditions
**  and observing wavelength.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     type   char[]   type of coordinates - "R", "H" or "A" (Notes 1,2)
**     ob1    double   observed Az, HA or RA (radians; Az is N=0,E=90)
**     ob2    double   observed ZD or Dec (radians)
**     utc1   double   UTC as a 2-part...
**     utc2   double   ...quasi Julian Date (Notes 3,4)
**     dut1   double   UT1-UTC (seconds, Note 5)
**     elong  double   longitude (radians, east +ve, Note 6)
**     phi    double   geodetic latitude (radians, Note 6)
**     hm     double   height above ellipsoid (m, geodetic Notes 6,8)
**     xp,yp  double   polar motion coordinates (radians, Note 7)
**     phpa   double   pressure at the observer (hPa = mB, Note 8)
**     tc     double   ambient temperature at the observer (deg C)
**     rh     double   relative humidity at the observer (range 0-1)
**     wl     double   wavelength (micrometers, Note 9)
**
**  Returned:
**     rc,dc  double   ICRS astrometric RA,Dec (radians)
**
**  Returned (function value):
**            int      status: +1 = dubious year (Note 4)
**                              0 = OK
**                             -1 = unacceptable date
**
**  Notes:
**
**  1)  "Observed" Az,ZD means the position that would be seen by a
**      perfect geodetically aligned theodolite.  (Zenith distance is
**      used rather than altitude in order to reflect the fact that no
**      allowance is made for depression of the horizon.)  This is
**      related to the observed HA,Dec via the standard rotation, using
**      the geodetic latitude (corrected for polar motion), while the
**      observed HA and (CIO-based) RA are related simply through the
**      Earth rotation angle and the site longitude.  "Observed" RA,Dec
**      or HA,Dec thus means the position that would be seen by a
**      perfect equatorial with its polar axis aligned to the Earth's
**      axis of rotation.
**
**  2)  Only the first character of the type argument is significant.
**      "R" or "r" indicates that ob1 and ob2 are the observed right
**      ascension (CIO-based) and declination;  "H" or "h" indicates
**      that they are hour angle (west +ve) and declination;  anything
**      else ("A" or "a" is recommended) indicates that ob1 and ob2 are
**      azimuth (north zero, east 90 deg) and zenith distance.
**
**  3)  utc1+utc2 is quasi Julian Date (see Note 2), apportioned in any
**      convenient way between the two arguments, for example where utc1
**      is the Julian Day Number and utc2 is the fraction of a day.
**
**      However, JD cannot unambiguously represent UTC during a leap
**      second unless special measures are taken.  The convention in the
**      present function is that the JD day represents UTC days whether
```

```
**        the length is 86399, 86400 or 86401 SI seconds.
**
**        Applications should use the function iauDtf2d to convert from
**        calendar date and time of day into 2-part quasi Julian Date, as
**        it implements the leap-second-ambiguity convention just
**        described.
**
**    4)  The warning status "dubious year" flags UTCs that predate the
**        introduction of the time scale or that are too far in the
**        future to be trusted.  See iauDat for further details.
**
**    5)  UT1-UTC is tabulated in IERS bulletins.  It increases by exactly
**        one second at the end of each positive UTC leap second,
**        introduced in order to keep UT1-UTC within +/- 0.9s.  n.b. This
**        practice is under review, and in the future UT1-UTC may grow
**        essentially without limit.
**
**    6)  The geographical coordinates are with respect to the WGS84
**        reference ellipsoid.  TAKE CARE WITH THE LONGITUDE SIGN:  the
**        longitude required by the present function is east-positive
**        (i.e. right-handed), in accordance with geographical convention.
**
**    7)  The polar motion xp,yp can be obtained from IERS bulletins.  The
**        values are the coordinates (in radians) of the Celestial
**        Intermediate Pole with respect to the International Terrestrial
**        Reference System (see IERS Conventions 2003), measured along the
**        meridians 0 and 90 deg west respectively.  For many
**        applications, xp and yp can be set to zero.
**
**    8)  If hm, the height above the ellipsoid of the observing station
**        in meters, is not known but phpa, the pressure in hPa (=mB), is
**        available, an adequate estimate of hm can be obtained from the
**        expression
**
**              hm = -29.3 * tsl * log ( phpa / 1013.25 );
**
**        where tsl is the approximate sea-level air temperature in K
**        (See Astrophysical Quantities, C.W.Allen, 3rd edition, section
**        52).  Similarly, if the pressure phpa is not known, it can be
**        estimated from the height of the observing station, hm, as
**        follows:
**
**              phpa = 1013.25 * exp ( -hm / ( 29.3 * tsl ) );
**
**        Note, however, that the refraction is nearly proportional to
**        the pressure and that an accurate phpa value is important for
**        precise work.
**
**    9)  The argument wl specifies the observing wavelength in
**        micrometers.  The transition from optical to radio is assumed to
**        occur at 100 micrometers (about 3000 GHz).
**
**    10) The accuracy of the result is limited by the corrections for
**        refraction, which use a simple A*tan(z) + B*tan^3(z) model.
**        Providing the meteorological parameters are known accurately and
**        there are no gross local effects, the predicted astrometric
**        coordinates should be within 0.05 arcsec (optical) or 1 arcsec
**        (radio) for a zenith distance of less than 70 degrees, better
**        than 30 arcsec (optical or radio) at 85 degrees and better
**        than 20 arcmin (optical) or 30 arcmin (radio) at the horizon.
**
**        Without refraction, the complementary functions iauAtco13 and
**        iauAtoc13 are self-consistent to better than 1 microarcsecond
**        all over the celestial sphere.  With refraction included,
**        consistency falls off at high zenith distances, but is still
**        better than 0.05 arcsec at 85 degrees.
**
**    11) It is advisable to take great care with units, as even unlikely
**        values of the input parameters are accepted and processed in
**        accordance with the models used.
**
**    Called:
**       iauApco13    astrometry parameters, ICRS-observed
```

```
**     iauAtoiq     quick observed to CIRS
**     iauAticq     quick CIRS to ICRS
**
*/
```

```
int iauAtoi13(const char *type, double ob1, double ob2,
              double utc1, double utc2, double dut1,
              double elong, double phi, double hm, double xp, double yp,
              double phpa, double tc, double rh, double wl,
              double *ri, double *di)
/*
**  - - - - - - - - - -
**   i a u A t o i 1 3
**  - - - - - - - - - -
**
**  Observed place to CIRS.  The caller supplies UTC, site coordinates,
**  ambient air conditions and observing wavelength.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     type   char[]   type of coordinates - "R", "H" or "A" (Notes 1,2)
**     ob1    double   observed Az, HA or RA (radians; Az is N=0,E=90)
**     ob2    double   observed ZD or Dec (radians)
**     utc1   double   UTC as a 2-part...
**     utc2   double   ...quasi Julian Date (Notes 3,4)
**     dut1   double   UT1-UTC (seconds, Note 5)
**     elong  double   longitude (radians, east +ve, Note 6)
**     phi    double   geodetic latitude (radians, Note 6)
**     hm     double   height above the ellipsoid (meters, Notes 6,8)
**     xp,yp  double   polar motion coordinates (radians, Note 7)
**     phpa   double   pressure at the observer (hPa = mB, Note 8)
**     tc     double   ambient temperature at the observer (deg C)
**     rh     double   relative humidity at the observer (range 0-1)
**     wl     double   wavelength (micrometers, Note 9)
**
**  Returned:
**     ri     double*  CIRS right ascension (CIO-based, radians)
**     di     double*  CIRS declination (radians)
**
**  Returned (function value):
**            int      status: +1 = dubious year (Note 2)
**                              0 = OK
**                             -1 = unacceptable date
**
**  Notes:
**
**  1)  "Observed" Az,ZD means the position that would be seen by a
**      perfect geodetically aligned theodolite.  (Zenith distance is
**      used rather than altitude in order to reflect the fact that no
**      allowance is made for depression of the horizon.)  This is
**      related to the observed HA,Dec via the standard rotation, using
**      the geodetic latitude (corrected for polar motion), while the
**      observed HA and (CIO-based) RA are related simply through the
**      Earth rotation angle and the site longitude.  "Observed" RA,Dec
**      or HA,Dec thus means the position that would be seen by a
**      perfect equatorial with its polar axis aligned to the Earth's
**      axis of rotation.
**
**  2)  Only the first character of the type argument is significant.
**      "R" or "r" indicates that ob1 and ob2 are the observed right
**      ascension and declination;  "H" or "h" indicates that they are
**      hour angle (west +ve) and declination;  anything else ("A" or
**      "a" is recommended) indicates that ob1 and ob2 are azimuth
**      (north zero, east 90 deg) and zenith distance.
**
**  3)  utc1+utc2 is quasi Julian Date (see Note 2), apportioned in any
**      convenient way between the two arguments, for example where utc1
**      is the Julian Day Number and utc2 is the fraction of a day.
**
**      However, JD cannot unambiguously represent UTC during a leap
**      second unless special measures are taken.  The convention in the
**      present function is that the JD day represents UTC days whether
```

```
**      the length is 86399, 86400 or 86401 SI seconds.
**
**      Applications should use the function iauDtf2d to convert from
**      calendar date and time of day into 2-part quasi Julian Date, as
**      it implements the leap-second-ambiguity convention just
**      described.
**
**  4)  The warning status "dubious year" flags UTCs that predate the
**      introduction of the time scale or that are too far in the
**      future to be trusted.  See iauDat for further details.
**
**  5)  UT1-UTC is tabulated in IERS bulletins.  It increases by exactly
**      one second at the end of each positive UTC leap second,
**      introduced in order to keep UT1-UTC within +/- 0.9s.  n.b. This
**      practice is under review, and in the future UT1-UTC may grow
**      essentially without limit.
**
**  6)  The geographical coordinates are with respect to the WGS84
**      reference ellipsoid.  TAKE CARE WITH THE LONGITUDE SIGN:  the
**      longitude required by the present function is east-positive
**      (i.e. right-handed), in accordance with geographical convention.
**
**  7)  The polar motion xp,yp can be obtained from IERS bulletins.  The
**      values are the coordinates (in radians) of the Celestial
**      Intermediate Pole with respect to the International Terrestrial
**      Reference System (see IERS Conventions 2003), measured along the
**      meridians 0 and 90 deg west respectively.  For many
**      applications, xp and yp can be set to zero.
**
**  8)  If hm, the height above the ellipsoid of the observing station
**      in meters, is not known but phpa, the pressure in hPa (=mB), is
**      available, an adequate estimate of hm can be obtained from the
**      expression
**
**            hm = -29.3 * tsl * log ( phpa / 1013.25 );
**
**      where tsl is the approximate sea-level air temperature in K
**      (See Astrophysical Quantities, C.W.Allen, 3rd edition, section
**      52).  Similarly, if the pressure phpa is not known, it can be
**      estimated from the height of the observing station, hm, as
**      follows:
**
**            phpa = 1013.25 * exp ( -hm / ( 29.3 * tsl ) );
**
**      Note, however, that the refraction is nearly proportional to
**      the pressure and that an accurate phpa value is important for
**      precise work.
**
**  9)  The argument wl specifies the observing wavelength in
**      micrometers.  The transition from optical to radio is assumed to
**      occur at 100 micrometers (about 3000 GHz).
**
**  10) The accuracy of the result is limited by the corrections for
**      refraction, which use a simple A*tan(z) + B*tan^3(z) model.
**      Providing the meteorological parameters are known accurately and
**      there are no gross local effects, the predicted astrometric
**      coordinates should be within 0.05 arcsec (optical) or 1 arcsec
**      (radio) for a zenith distance of less than 70 degrees, better
**      than 30 arcsec (optical or radio) at 85 degrees and better
**      than 20 arcmin (optical) or 30 arcmin (radio) at the horizon.
**
**      Without refraction, the complementary functions iauAtio13 and
**      iauAtoi13 are self-consistent to better than 1 microarcsecond
**      all over the celestial sphere.  With refraction included,
**      consistency falls off at high zenith distances, but is still
**      better than 0.05 arcsec at 85 degrees.
**
**  12) It is advisable to take great care with units, as even unlikely
**      values of the input parameters are accepted and processed in
**      accordance with the models used.
**
**  Called:
**     iauApio13    astrometry parameters, CIRS-observed, 2013
```

```
**      iauAtoiq     quick observed to CIRS
**
*/
```

```
void iauAtoiq(const char *type,
              double ob1, double ob2, iauASTROM *astrom,
              double *ri, double *di)
/*
**  - - - - - - - - -
**   i a u A t o i q
**  - - - - - - - - -
**
**  Quick observed place to CIRS, given the star-independent astrometry
**  parameters.
**
**  Use of this function is appropriate when efficiency is important and
**  where many star positions are all to be transformed for one date.
**  The star-independent astrometry parameters can be obtained by
**  calling iauApio[13] or iauApco[13].
**
**  Status:  support function.
**
**  Given:
**     type   char[]     type of coordinates: "R", "H" or "A" (Note 1)
**     ob1    double     observed Az, HA or RA (radians; Az is N=0,E=90)
**     ob2    double     observed ZD or Dec (radians)
**     astrom iauASTROM* star-independent astrometry parameters:
**      pmt    double       PM time interval (SSB, Julian years)
**      eb     double[3]    SSB to observer (vector, au)
**      eh     double[3]    Sun to observer (unit vector)
**      em     double       distance from Sun to observer (au)
**      v      double[3]    barycentric observer velocity (vector, c)
**      bm1    double       sqrt(1-|v|^2): reciprocal of Lorenz factor
**      bpn    double[3][3] bias-precession-nutation matrix
**      along  double       longitude + s' (radians)
**      xpl    double       polar motion xp wrt local meridian (radians)
**      ypl    double       polar motion yp wrt local meridian (radians)
**      sphi   double       sine of geodetic latitude
**      cphi   double       cosine of geodetic latitude
**      diurab double       magnitude of diurnal aberration vector
**      eral   double       "local" Earth rotation angle (radians)
**      refa   double       refraction constant A (radians)
**      refb   double       refraction constant B (radians)
**
**  Returned:
**     ri     double*    CIRS right ascension (CIO-based, radians)
**     di     double*    CIRS declination (radians)
**
**  Notes:
**
**  1) "Observed" Az,ZD means the position that would be seen by a
**     perfect geodetically aligned theodolite.  This is related to
**     the observed HA,Dec via the standard rotation, using the geodetic
**     latitude (corrected for polar motion), while the observed HA and
**     (CIO-based) RA are related simply through the Earth rotation
**     angle and the site longitude.  "Observed" RA,Dec or HA,Dec thus
**     means the position that would be seen by a perfect equatorial
**     with its polar axis aligned to the Earth's axis of rotation.
**
**  2) Only the first character of the type argument is significant.
**     "R" or "r" indicates that ob1 and ob2 are the observed right
**     ascension (CIO-based) and declination;  "H" or "h" indicates that
**     they are hour angle (west +ve) and declination;  anything else
**     ("A" or "a" is recommended) indicates that ob1 and ob2 are
**     azimuth (north zero, east 90 deg) and zenith distance.  (Zenith
**     distance is used rather than altitude in order to reflect the
**     fact that no allowance is made for depression of the horizon.)
**
**  3) The accuracy of the result is limited by the corrections for
**     refraction, which use a simple A*tan(z) + B*tan^3(z) model.
**     Providing the meteorological parameters are known accurately and
**     there are no gross local effects, the predicted intermediate
**     coordinates should be within 0.05 arcsec (optical) or 1 arcsec
**     (radio) for a zenith distance of less than 70 degrees, better
**     than 30 arcsec (optical or radio) at 85 degrees and better than
```

```
**     20 arcmin (optical) or 25 arcmin (radio) at the horizon.
**
**     Without refraction, the complementary functions iauAtioq and
**     iauAtoiq are self-consistent to better than 1 microarcsecond all
**     over the celestial sphere.  With refraction included, consistency
**     falls off at high zenith distances, but is still better than
**     0.05 arcsec at 85 degrees.
**
**  4) It is advisable to take great care with units, as even unlikely
**     values of the input parameters are accepted and processed in
**     accordance with the models used.
**
**  Called:
**     iauS2c       spherical coordinates to unit vector
**     iauC2s       p-vector to spherical
**     iauAnp       normalize angle into range 0 to 2pi
**
*/
```

```
void iauBi00(double *dpsibi, double *depsbi, double *dra)
/*
**  - - - - - - - -
**   i a u B i 0 0
**  - - - - - - - -
**
**  Frame bias components of IAU 2000 precession-nutation models;  part
**  of the Mathews-Herring-Buffett (MHB2000) nutation series, with
**  additions.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical model.
**
**  Returned:
**     dpsibi,depsbi  double  longitude and obliquity corrections
**     dra            double  the ICRS RA of the J2000.0 mean equinox
**
**  Notes:
**
**  1) The frame bias corrections in longitude and obliquity (radians)
**     are required in order to correct for the offset between the GCRS
**     pole and the mean J2000.0 pole.  They define, with respect to the
**     GCRS frame, a J2000.0 mean pole that is consistent with the rest
**     of the IAU 2000A precession-nutation model.
**
**  2) In addition to the displacement of the pole, the complete
**     description of the frame bias requires also an offset in right
**     ascension.  This is not part of the IAU 2000A model, and is from
**     Chapront et al. (2002).  It is returned in radians.
**
**  3) This is a supplemented implementation of one aspect of the IAU
**     2000A nutation model, formally adopted by the IAU General
**     Assembly in 2000, namely MHB2000 (Mathews et al. 2002).
**
**  References:
**
**     Chapront, J., Chapront-Touze, M. & Francou, G., Astron.
**     Astrophys., 387, 700, 2002.
**
**     Mathews, P.M., Herring, T.A., Buffet, B.A., "Modeling of nutation
**     and precession:  New nutation series for nonrigid Earth and
**     insights into the Earth's interior", J.Geophys.Res., 107, B4,
**     2002.  The MHB2000 code itself was obtained on 2002 September 9
**     from ftp://maia.usno.navy.mil/conv2000/chapter5/IAU2000A.
**
*/
```

```
void iauBp00(double date1, double date2,
             double rb[3][3], double rp[3][3], double rbp[3][3])
/*
**  - - - - - - - -
**   i a u B p 0 0
**  - - - - - - - -
**
**  Frame bias and precession, IAU 2000.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical model.
**
**  Given:
**     date1,date2  double         TT as a 2-part Julian Date (Note 1)
**
**  Returned:
**     rb           double[3][3]   frame bias matrix (Note 2)
**     rp           double[3][3]   precession matrix (Note 3)
**     rbp          double[3][3]   bias-precession matrix (Note 4)
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**             date1         date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The matrix rb transforms vectors from GCRS to mean J2000.0 by
**     applying frame bias.
**
**  3) The matrix rp transforms vectors from J2000.0 mean equator and
**     equinox to mean equator and equinox of date by applying
**     precession.
**
**  4) The matrix rbp transforms vectors from GCRS to mean equator and
**     equinox of date by applying frame bias then precession.  It is
**     the product rp x rb.
**
**  5) It is permissible to re-use the same array in the returned
**     arguments.  The arrays are filled in the order given.
**
**  Called:
**     iauBi00      frame bias components, IAU 2000
**     iauPr00      IAU 2000 precession adjustments
**     iauIr        initialize r-matrix to identity
**     iauRx        rotate around X-axis
**     iauRy        rotate around Y-axis
**     iauRz        rotate around Z-axis
**     iauCr        copy r-matrix
**     iauRxr       product of two r-matrices
**
**  Reference:
**     "Expressions for the Celestial Intermediate Pole and Celestial
**     Ephemeris Origin consistent with the IAU 2000A precession-
**     nutation model", Astron.Astrophys. 400, 1145-1154 (2003)
```

```
**
**     n.b. The celestial ephemeris origin (CEO) was renamed "celestial
**          intermediate origin" (CIO) by IAU 2006 Resolution 2.
**
*/
```

```
void iauBp06(double date1, double date2,
             double rb[3][3], double rp[3][3], double rbp[3][3])
/*
**  - - - - - - - -
**   i a u B p 0 6
**  - - - - - - - -
**
**  Frame bias and precession, IAU 2006.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1,date2  double         TT as a 2-part Julian Date (Note 1)
**
**  Returned:
**     rb           double[3][3]   frame bias matrix (Note 2)
**     rp           double[3][3]   precession matrix (Note 3)
**     rbp          double[3][3]   bias-precession matrix (Note 4)
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**             date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The matrix rb transforms vectors from GCRS to mean J2000.0 by
**     applying frame bias.
**
**  3) The matrix rp transforms vectors from mean J2000.0 to mean of
**     date by applying precession.
**
**  4) The matrix rbp transforms vectors from GCRS to mean of date by
**     applying frame bias then precession.  It is the product rp x rb.
**
**  5) It is permissible to re-use the same array in the returned
**     arguments.  The arrays are filled in the order given.
**
**  Called:
**     iauPfw06     bias-precession F-W angles, IAU 2006
**     iauFw2m      F-W angles to r-matrix
**     iauPmat06    PB matrix, IAU 2006
**     iauTr        transpose r-matrix
**     iauRxr       product of two r-matrices
**     iauCr        copy r-matrix
**
**  References:
**
**     Capitaine, N. & Wallace, P.T., 2006, Astron.Astrophys. 450, 855
**
**     Wallace, P.T. & Capitaine, N., 2006, Astron.Astrophys. 459, 981
**
*/
```

```
void iauBpn2xy(double rbpn[3][3], double *x, double *y)
/*
**  - - - - - - - - - - 
**   i a u B p n 2 x y
**  - - - - - - - - - -
**
**  Extract from the bias-precession-nutation matrix the X,Y coordinates
**  of the Celestial Intermediate Pole.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     rbpn      double[3][3]  celestial-to-true matrix (Note 1)
**
**  Returned:
**     x,y       double        Celestial Intermediate Pole (Note 2)
**
**  Notes:
**
**  1) The matrix rbpn transforms vectors from GCRS to true equator (and
**     CIO or equinox) of date, and therefore the Celestial Intermediate
**     Pole unit vector is the bottom row of the matrix.
**
**  2) The arguments x,y are components of the Celestial Intermediate
**     Pole unit vector in the Geocentric Celestial Reference System.
**
**  Reference:
**
**     "Expressions for the Celestial Intermediate Pole and Celestial
**     Ephemeris Origin consistent with the IAU 2000A precession-
**     nutation model", Astron.Astrophys. 400, 1145-1154
**     (2003)
**
**     n.b. The celestial ephemeris origin (CEO) was renamed "celestial
**          intermediate origin" (CIO) by IAU 2006 Resolution 2.
**
*/
```

```
void iauC2i00a(double date1, double date2, double rc2i[3][3])
/*
**  - - - - - - - - - -
**   i a u C 2 i 0 0 a
**  - - - - - - - - - -
**
**  Form the celestial-to-intermediate matrix for a given date using the
**  IAU 2000A precession-nutation model.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1,date2 double       TT as a 2-part Julian Date (Note 1)
**
**  Returned:
**     rc2i        double[3][3] celestial-to-intermediate matrix (Note 2)
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The matrix rc2i is the first stage in the transformation from
**     celestial to terrestrial coordinates:
**
**        [TRS]  =  RPOM * R_3(ERA) * rc2i * [CRS]
**
**               =  rc2t * [CRS]
**
**     where [CRS] is a vector in the Geocentric Celestial Reference
**     System and [TRS] is a vector in the International Terrestrial
**     Reference System (see IERS Conventions 2003), ERA is the Earth
**     Rotation Angle and RPOM is the polar motion matrix.
**
**  3) A faster, but slightly less accurate, result (about 1 mas) can be
**     obtained by using instead the iauC2i00b function.
**
**  Called:
**     iauPnm00a    classical NPB matrix, IAU 2000A
**     iauC2ibpn    celestial-to-intermediate matrix, given NPB matrix
**
**  References:
**
**     "Expressions for the Celestial Intermediate Pole and Celestial
**     Ephemeris Origin consistent with the IAU 2000A precession-
**     nutation model", Astron.Astrophys. 400, 1145-1154
**     (2003)
**
**     n.b. The celestial ephemeris origin (CEO) was renamed "celestial
**          intermediate origin" (CIO) by IAU 2006 Resolution 2.
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
```

```
**      IERS Technical Note No. 32, BKG (2004)
**
*/
```

```
void iauC2i00b(double date1, double date2, double rc2i[3][3])
/*
**  - - - - - - - - - -
**   i a u C 2 i 0 0 b
**  - - - - - - - - - -
**
**  Form the celestial-to-intermediate matrix for a given date using the
**  IAU 2000B precession-nutation model.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1,date2 double       TT as a 2-part Julian Date (Note 1)
**
**  Returned:
**     rc2i        double[3][3] celestial-to-intermediate matrix (Note 2)
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The matrix rc2i is the first stage in the transformation from
**     celestial to terrestrial coordinates:
**
**        [TRS]  =  RPOM * R_3(ERA) * rc2i * [CRS]
**
**               =  rc2t * [CRS]
**
**     where [CRS] is a vector in the Geocentric Celestial Reference
**     System and [TRS] is a vector in the International Terrestrial
**     Reference System (see IERS Conventions 2003), ERA is the Earth
**     Rotation Angle and RPOM is the polar motion matrix.
**
**  3) The present function is faster, but slightly less accurate (about
**     1 mas), than the iauC2i00a function.
**
**  Called:
**     iauPnm00b    classical NPB matrix, IAU 2000B
**     iauC2ibpn    celestial-to-intermediate matrix, given NPB matrix
**
**  References:
**
**     "Expressions for the Celestial Intermediate Pole and Celestial
**     Ephemeris Origin consistent with the IAU 2000A precession-
**     nutation model", Astron.Astrophys. 400, 1145-1154
**     (2003)
**
**     n.b. The celestial ephemeris origin (CEO) was renamed "celestial
**          intermediate origin" (CIO) by IAU 2006 Resolution 2.
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
```

```
**      IERS Technical Note No. 32, BKG (2004)
**
*/
```

```
void iauC2i06a(double date1, double date2, double rc2i[3][3])
/*
**  - - - - - - - - - -
**   i a u C 2 i 0 6 a
**  - - - - - - - - - -
**
**  Form the celestial-to-intermediate matrix for a given date using the
**  IAU 2006 precession and IAU 2000A nutation models.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1,date2 double       TT as a 2-part Julian Date (Note 1)
**
**  Returned:
**     rc2i        double[3][3] celestial-to-intermediate matrix (Note 2)
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The matrix rc2i is the first stage in the transformation from
**     celestial to terrestrial coordinates:
**
**        [TRS]  =  RPOM * R_3(ERA) * rc2i * [CRS]
**
**               =  RC2T * [CRS]
**
**     where [CRS] is a vector in the Geocentric Celestial Reference
**     System and [TRS] is a vector in the International Terrestrial
**     Reference System (see IERS Conventions 2003), ERA is the Earth
**     Rotation Angle and RPOM is the polar motion matrix.
**
**  Called:
**     iauPnm06a    classical NPB matrix, IAU 2006/2000A
**     iauBpn2xy    extract CIP X,Y coordinates from NPB matrix
**     iauS06       the CIO locator s, given X,Y, IAU 2006
**     iauC2ixys    celestial-to-intermediate matrix, given X,Y and s
**
**  References:
**
**     McCarthy, D. D., Petit, G. (eds.), 2004, IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG
**
*/
```

```
void iauC2ibpn(double date1, double date2, double rbpn[3][3],
               double rc2i[3][3])
/*
**  - - - - - - - - - -
**   i a u C 2 i b p n
**  - - - - - - - - - -
**
**  Form the celestial-to-intermediate matrix for a given date given
**  the bias-precession-nutation matrix.  IAU 2000.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1,date2 double       TT as a 2-part Julian Date (Note 1)
**     rbpn        double[3][3] celestial-to-true matrix (Note 2)
**
**  Returned:
**     rc2i        double[3][3] celestial-to-intermediate matrix (Note 3)
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The matrix rbpn transforms vectors from GCRS to true equator (and
**     CIO or equinox) of date.  Only the CIP (bottom row) is used.
**
**  3) The matrix rc2i is the first stage in the transformation from
**     celestial to terrestrial coordinates:
**
**        [TRS] = RPOM * R_3(ERA) * rc2i * [CRS]
**
**              = RC2T * [CRS]
**
**     where [CRS] is a vector in the Geocentric Celestial Reference
**     System and [TRS] is a vector in the International Terrestrial
**     Reference System (see IERS Conventions 2003), ERA is the Earth
**     Rotation Angle and RPOM is the polar motion matrix.
**
**  4) Although its name does not include "00", This function is in fact
**     specific to the IAU 2000 models.
**
**  Called:
**     iauBpn2xy    extract CIP X,Y coordinates from NPB matrix
**     iauC2ixy     celestial-to-intermediate matrix, given X,Y
**
**  References:
**     "Expressions for the Celestial Intermediate Pole and Celestial
**     Ephemeris Origin consistent with the IAU 2000A precession-
**     nutation model", Astron.Astrophys. 400, 1145-1154 (2003)
**
**     n.b. The celestial ephemeris origin (CEO) was renamed "celestial
```

```
**          intermediate origin" (CIO) by IAU 2006 Resolution 2.
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG (2004)
**
*/
```

```
void iauC2ixy(double date1, double date2, double x, double y,
              double rc2i[3][3])
/*
**  - - - - - - - - - -
**   i a u C 2 i x y
**  - - - - - - - - - -
**
**  Form the celestial to intermediate-frame-of-date matrix for a given
**  date when the CIP X,Y coordinates are known.  IAU 2000.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1,date2 double       TT as a 2-part Julian Date (Note 1)
**     x,y         double       Celestial Intermediate Pole (Note 2)
**
**  Returned:
**     rc2i        double[3][3] celestial-to-intermediate matrix (Note 3)
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The Celestial Intermediate Pole coordinates are the x,y components
**     of the unit vector in the Geocentric Celestial Reference System.
**
**  3) The matrix rc2i is the first stage in the transformation from
**     celestial to terrestrial coordinates:
**
**        [TRS] = RPOM * R_3(ERA) * rc2i * [CRS]
**
**              = RC2T * [CRS]
**
**     where [CRS] is a vector in the Geocentric Celestial Reference
**     System and [TRS] is a vector in the International Terrestrial
**     Reference System (see IERS Conventions 2003), ERA is the Earth
**     Rotation Angle and RPOM is the polar motion matrix.
**
**  4) Although its name does not include "00", This function is in fact
**     specific to the IAU 2000 models.
**
**  Called:
**     iauC2ixys    celestial-to-intermediate matrix, given X,Y and s
**     iauS00       the CIO locator s, given X,Y, IAU 2000A
**
**  Reference:
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG (2004)
**
*/
```

```
void iauC2ixys(double x, double y, double s, double rc2i[3][3])
/*
**  - - - - - - - - - -
**   i a u C 2 i x y s
**  - - - - - - - - - -
**
**  Form the celestial to intermediate-frame-of-date matrix given the CIP
**  X,Y and the CIO locator s.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     x,y      double         Celestial Intermediate Pole (Note 1)
**     s        double         the CIO locator s (Note 2)
**
**  Returned:
**     rc2i     double[3][3]   celestial-to-intermediate matrix (Note 3)
**
**  Notes:
**
**  1) The Celestial Intermediate Pole coordinates are the x,y
**     components of the unit vector in the Geocentric Celestial
**     Reference System.
**
**  2) The CIO locator s (in radians) positions the Celestial
**     Intermediate Origin on the equator of the CIP.
**
**  3) The matrix rc2i is the first stage in the transformation from
**     celestial to terrestrial coordinates:
**
**        [TRS] = RPOM * R_3(ERA) * rc2i * [CRS]
**
**              = RC2T * [CRS]
**
**     where [CRS] is a vector in the Geocentric Celestial Reference
**     System and [TRS] is a vector in the International Terrestrial
**     Reference System (see IERS Conventions 2003), ERA is the Earth
**     Rotation Angle and RPOM is the polar motion matrix.
**
**  Called:
**     iauIr        initialize r-matrix to identity
**     iauRz        rotate around Z-axis
**     iauRy        rotate around Y-axis
**
**  Reference:
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG (2004)
**
*/
```

```
void iauC2s(double p[3], double *theta, double *phi)
/*
**  - - - - - - - -
**   i a u C 2 s
**  - - - - - - - -
**
**  P-vector to spherical coordinates.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     p      double[3]    p-vector
**
**  Returned:
**     theta  double       longitude angle (radians)
**     phi    double       latitude angle (radians)
**
**  Notes:
**
**  1) The vector p can have any magnitude; only its direction is used.
**
**  2) If p is null, zero theta and phi are returned.
**
**  3) At either pole, zero theta is returned.
**
*/
```

```
void iauC2t00a(double tta, double ttb, double uta, double utb,
               double xp, double yp, double rc2t[3][3])
/*
**  - - - - - - - - - -
**   i a u C 2 t 0 0 a
**  - - - - - - - - - -
**
**  Form the celestial to terrestrial matrix given the date, the UT1 and
**  the polar motion, using the IAU 2000A precession-nutation model.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     tta,ttb  double         TT as a 2-part Julian Date (Note 1)
**     uta,utb  double         UT1 as a 2-part Julian Date (Note 1)
**     xp,yp    double         CIP coordinates (radians, Note 2)
**
**  Returned:
**     rc2t     double[3][3]   celestial-to-terrestrial matrix (Note 3)
**
**  Notes:
**
**  1) The TT and UT1 dates tta+ttb and uta+utb are Julian Dates,
**     apportioned in any convenient way between the arguments uta and
**     utb.  For example, JD(UT1)=2450123.7 could be expressed in any of
**     these ways, among others:
**
**             uta            utb
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution is
**     acceptable.  The J2000 and MJD methods are good compromises
**     between resolution and convenience.  In the case of uta,utb, the
**     date & time method is best matched to the Earth rotation angle
**     algorithm used:  maximum precision is delivered when the uta
**     argument is for 0hrs UT1 on the day in question and the utb
**     argument lies in the range 0 to 1, or vice versa.
**
**  2) The arguments xp and yp are the coordinates (in radians) of the
**     Celestial Intermediate Pole with respect to the International
**     Terrestrial Reference System (see IERS Conventions 2003),
**     measured along the meridians 0 and 90 deg west respectively.
**
**  3) The matrix rc2t transforms from celestial to terrestrial
**     coordinates:
**
**        [TRS] = RPOM * R_3(ERA) * RC2I * [CRS]
**
**              = rc2t * [CRS]
**
**     where [CRS] is a vector in the Geocentric Celestial Reference
**     System and [TRS] is a vector in the International Terrestrial
**     Reference System (see IERS Conventions 2003), RC2I is the
**     celestial-to-intermediate matrix, ERA is the Earth rotation
**     angle and RPOM is the polar motion matrix.
**
**  4) A faster, but slightly less accurate, result (about 1 mas) can
**     be obtained by using instead the iauC2t00b function.
**
**  Called:
**     iauC2i00a    celestial-to-intermediate matrix, IAU 2000A
**     iauEra00     Earth rotation angle, IAU 2000
**     iauSp00      the TIO locator s', IERS 2000
```

```
**     iauPom00     polar motion matrix
**     iauC2tcio    form CIO-based celestial-to-terrestrial matrix
**
**  Reference:
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG (2004)
**
*/
```

```
void iauC2t00b(double tta, double ttb, double uta, double utb,
               double xp, double yp, double rc2t[3][3])
/*
**  - - - - - - - - - -
**   i a u C 2 t 0 0 b
**  - - - - - - - - - -
**
**  Form the celestial to terrestrial matrix given the date, the UT1 and
**  the polar motion, using the IAU 2000B precession-nutation model.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     tta,ttb  double         TT as a 2-part Julian Date (Note 1)
**     uta,utb  double         UT1 as a 2-part Julian Date (Note 1)
**     xp,yp    double         coordinates of the pole (radians, Note 2)
**
**  Returned:
**     rc2t     double[3][3]   celestial-to-terrestrial matrix (Note 3)
**
**  Notes:
**
**  1) The TT and UT1 dates tta+ttb and uta+utb are Julian Dates,
**     apportioned in any convenient way between the arguments uta and
**     utb.  For example, JD(UT1)=2450123.7 could be expressed in any of
**     these ways, among others:
**
**             uta            utb
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution is
**     acceptable.  The J2000 and MJD methods are good compromises
**     between resolution and convenience.  In the case of uta,utb, the
**     date & time method is best matched to the Earth rotation angle
**     algorithm used:  maximum precision is delivered when the uta
**     argument is for 0hrs UT1 on the day in question and the utb
**     argument lies in the range 0 to 1, or vice versa.
**
**  2) The arguments xp and yp are the coordinates (in radians) of the
**     Celestial Intermediate Pole with respect to the International
**     Terrestrial Reference System (see IERS Conventions 2003),
**     measured along the meridians 0 and 90 deg west respectively.
**
**  3) The matrix rc2t transforms from celestial to terrestrial
**     coordinates:
**
**        [TRS] = RPOM * R_3(ERA) * RC2I * [CRS]
**
**              = rc2t * [CRS]
**
**     where [CRS] is a vector in the Geocentric Celestial Reference
**     System and [TRS] is a vector in the International Terrestrial
**     Reference System (see IERS Conventions 2003), RC2I is the
**     celestial-to-intermediate matrix, ERA is the Earth rotation
**     angle and RPOM is the polar motion matrix.
**
**  4) The present function is faster, but slightly less accurate (about
**     1 mas), than the iauC2t00a function.
**
**  Called:
**     iauC2i00b    celestial-to-intermediate matrix, IAU 2000B
**     iauEra00     Earth rotation angle, IAU 2000
**     iauPom00     polar motion matrix
```

```
**     iauC2tcio    form CIO-based celestial-to-terrestrial matrix
**
**  Reference:
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG (2004)
**
*/
```

```
void iauC2t06a(double tta, double ttb, double uta, double utb,
               double xp, double yp, double rc2t[3][3])
/*
**  - - - - - - - - - -
**   i a u C 2 t 0 6 a
**  - - - - - - - - - -
**
**  Form the celestial to terrestrial matrix given the date, the UT1 and
**  the polar motion, using the IAU 2006/2000A precession-nutation
**  model.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     tta,ttb  double         TT as a 2-part Julian Date (Note 1)
**     uta,utb  double         UT1 as a 2-part Julian Date (Note 1)
**     xp,yp    double         coordinates of the pole (radians, Note 2)
**
**  Returned:
**     rc2t     double[3][3]   celestial-to-terrestrial matrix (Note 3)
**
**  Notes:
**
**  1) The TT and UT1 dates tta+ttb and uta+utb are Julian Dates,
**     apportioned in any convenient way between the two arguments.  For
**     example, JD(UT1)=2450123.7 could be expressed in any of
**     these ways, among others:
**
**             uta            utb
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution is
**     acceptable.  The J2000 and MJD methods are good compromises
**     between resolution and convenience.  In the case of uta,utb, the
**     date & time method is best matched to the Earth rotation angle
**     algorithm used:  maximum precision is delivered when the uta
**     argument is for 0hrs UT1 on the day in question and the utb
**     argument lies in the range 0 to 1, or vice versa.
**
**  2) The arguments xp and yp are the coordinates (in radians) of the
**     Celestial Intermediate Pole with respect to the International
**     Terrestrial Reference System (see IERS Conventions 2003),
**     measured along the meridians 0 and 90 deg west respectively.
**
**  3) The matrix rc2t transforms from celestial to terrestrial
**     coordinates:
**
**        [TRS] = RPOM * R_3(ERA) * RC2I * [CRS]
**
**              = rc2t * [CRS]
**
**     where [CRS] is a vector in the Geocentric Celestial Reference
**     System and [TRS] is a vector in the International Terrestrial
**     Reference System (see IERS Conventions 2003), RC2I is the
**     celestial-to-intermediate matrix, ERA is the Earth rotation
**     angle and RPOM is the polar motion matrix.
**
**  Called:
**     iauC2i06a    celestial-to-intermediate matrix, IAU 2006/2000A
**     iauEra00     Earth rotation angle, IAU 2000
**     iauSp00      the TIO locator s', IERS 2000
**     iauPom00     polar motion matrix
**     iauC2tcio    form CIO-based celestial-to-terrestrial matrix
```

```
**
**  Reference:
**
**     McCarthy, D. D., Petit, G. (eds.), 2004, IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG
**
*/
```

```
void iauC2tcio(double rc2i[3][3], double era, double rpom[3][3],
               double rc2t[3][3])
/*
**  - - - - - - - - - -
**   i a u C 2 t c i o
**  - - - - - - - - - -
**
**  Assemble the celestial to terrestrial matrix from CIO-based
**  components (the celestial-to-intermediate matrix, the Earth Rotation
**  Angle and the polar motion matrix).
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     rc2i     double[3][3]    celestial-to-intermediate matrix
**     era      double          Earth rotation angle (radians)
**     rpom     double[3][3]    polar-motion matrix
**
**  Returned:
**     rc2t     double[3][3]    celestial-to-terrestrial matrix
**
**  Notes:
**
**  1) This function constructs the rotation matrix that transforms
**     vectors in the celestial system into vectors in the terrestrial
**     system.  It does so starting from precomputed components, namely
**     the matrix which rotates from celestial coordinates to the
**     intermediate frame, the Earth rotation angle and the polar motion
**     matrix.  One use of the present function is when generating a
**     series of celestial-to-terrestrial matrices where only the Earth
**     Rotation Angle changes, avoiding the considerable overhead of
**     recomputing the precession-nutation more often than necessary to
**     achieve given accuracy objectives.
**
**  2) The relationship between the arguments is as follows:
**
**        [TRS] = RPOM * R_3(ERA) * rc2i * [CRS]
**
**              = rc2t * [CRS]
**
**     where [CRS] is a vector in the Geocentric Celestial Reference
**     System and [TRS] is a vector in the International Terrestrial
**     Reference System (see IERS Conventions 2003).
**
**  Called:
**     iauCr        copy r-matrix
**     iauRz        rotate around Z-axis
**     iauRxr       product of two r-matrices
**
**  Reference:
**
**     McCarthy, D. D., Petit, G. (eds.), 2004, IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG
**
*/
```

```
void iauC2teqx(double rbpn[3][3], double gst, double rpom[3][3],
               double rc2t[3][3])
/*
**  - - - - - - - - - -
**   i a u C 2 t e q x
**  - - - - - - - - - -
**
**  Assemble the celestial to terrestrial matrix from equinox-based
**  components (the celestial-to-true matrix, the Greenwich Apparent
**  Sidereal Time and the polar motion matrix).
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     rbpn   double[3][3]  celestial-to-true matrix
**     gst    double        Greenwich (apparent) Sidereal Time (radians)
**     rpom   double[3][3]  polar-motion matrix
**
**  Returned:
**     rc2t   double[3][3]  celestial-to-terrestrial matrix (Note 2)
**
**  Notes:
**
**  1) This function constructs the rotation matrix that transforms
**     vectors in the celestial system into vectors in the terrestrial
**     system.  It does so starting from precomputed components, namely
**     the matrix which rotates from celestial coordinates to the
**     true equator and equinox of date, the Greenwich Apparent Sidereal
**     Time and the polar motion matrix.  One use of the present function
**     is when generating a series of celestial-to-terrestrial matrices
**     where only the Sidereal Time changes, avoiding the considerable
**     overhead of recomputing the precession-nutation more often than
**     necessary to achieve given accuracy objectives.
**
**  2) The relationship between the arguments is as follows:
**
**        [TRS] = rpom * R_3(gst) * rbpn * [CRS]
**
**              = rc2t * [CRS]
**
**     where [CRS] is a vector in the Geocentric Celestial Reference
**     System and [TRS] is a vector in the International Terrestrial
**     Reference System (see IERS Conventions 2003).
**
**  Called:
**     iauCr        copy r-matrix
**     iauRz        rotate around Z-axis
**     iauRxr       product of two r-matrices
**
**  Reference:
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG (2004)
**
*/
```

```
void iauC2tpe(double tta, double ttb, double uta, double utb,
              double dpsi, double deps, double xp, double yp,
              double rc2t[3][3])
/*
**  - - - - - - - - -
**   i a u C 2 t p e
**  - - - - - - - - -
**
**  Form the celestial to terrestrial matrix given the date, the UT1,
**  the nutation and the polar motion.  IAU 2000.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     tta,ttb    double        TT as a 2-part Julian Date (Note 1)
**     uta,utb    double        UT1 as a 2-part Julian Date (Note 1)
**     dpsi,deps  double        nutation (Note 2)
**     xp,yp      double        coordinates of the pole (radians, Note 3)
**
**  Returned:
**     rc2t       double[3][3]  celestial-to-terrestrial matrix (Note 4)
**
**  Notes:
**
**  1) The TT and UT1 dates tta+ttb and uta+utb are Julian Dates,
**     apportioned in any convenient way between the arguments uta and
**     utb.  For example, JD(UT1)=2450123.7 could be expressed in any of
**     these ways, among others:
**
**             uta            utb
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution is
**     acceptable.  The J2000 and MJD methods are good compromises
**     between resolution and convenience.  In the case of uta,utb, the
**     date & time method is best matched to the Earth rotation angle
**     algorithm used:  maximum precision is delivered when the uta
**     argument is for 0hrs UT1 on the day in question and the utb
**     argument lies in the range 0 to 1, or vice versa.
**
**  2) The caller is responsible for providing the nutation components;
**     they are in longitude and obliquity, in radians and are with
**     respect to the equinox and ecliptic of date.  For high-accuracy
**     applications, free core nutation should be included as well as
**     any other relevant corrections to the position of the CIP.
**
**  3) The arguments xp and yp are the coordinates (in radians) of the
**     Celestial Intermediate Pole with respect to the International
**     Terrestrial Reference System (see IERS Conventions 2003),
**     measured along the meridians 0 and 90 deg west respectively.
**
**  4) The matrix rc2t transforms from celestial to terrestrial
**     coordinates:
**
**        [TRS] = RPOM * R_3(GST) * RBPN * [CRS]
**
**              = rc2t * [CRS]
**
**     where [CRS] is a vector in the Geocentric Celestial Reference
**     System and [TRS] is a vector in the International Terrestrial
**     Reference System (see IERS Conventions 2003), RBPN is the
**     bias-precession-nutation matrix, GST is the Greenwich (apparent)
**     Sidereal Time and RPOM is the polar motion matrix.
```

```
**
**  5) Although its name does not include "00", This function is in fact
**     specific to the IAU 2000 models.
**
**  Called:
**     iauPn00      bias/precession/nutation results, IAU 2000
**     iauGmst00    Greenwich mean sidereal time, IAU 2000
**     iauSp00      the TIO locator s', IERS 2000
**     iauEe00      equation of the equinoxes, IAU 2000
**     iauPom00     polar motion matrix
**     iauC2teqx    form equinox-based celestial-to-terrestrial matrix
**
**  Reference:
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG (2004)
**
*/
```

```
void iauC2txy(double tta, double ttb, double uta, double utb,
              double x, double y, double xp, double yp,
              double rc2t[3][3])
/*
**  - - - - - - - - -
**   i a u C 2 t x y
**  - - - - - - - - -
**
**  Form the celestial to terrestrial matrix given the date, the UT1,
**  the CIP coordinates and the polar motion.  IAU 2000.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     tta,ttb  double         TT as a 2-part Julian Date (Note 1)
**     uta,utb  double         UT1 as a 2-part Julian Date (Note 1)
**     x,y      double         Celestial Intermediate Pole (Note 2)
**     xp,yp    double         coordinates of the pole (radians, Note 3)
**
**  Returned:
**     rc2t     double[3][3]   celestial-to-terrestrial matrix (Note 4)
**
**  Notes:
**
**  1) The TT and UT1 dates tta+ttb and uta+utb are Julian Dates,
**     apportioned in any convenient way between the arguments uta and
**     utb.  For example, JD(UT1)=2450123.7 could be expressed in any o
**     these ways, among others:
**
**             uta            utb
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution is
**     acceptable.  The J2000 and MJD methods are good compromises
**     between resolution and convenience.  In the case of uta,utb, the
**     date & time method is best matched to the Earth rotation angle
**     algorithm used:  maximum precision is delivered when the uta
**     argument is for 0hrs UT1 on the day in question and the utb
**     argument lies in the range 0 to 1, or vice versa.
**
**  2) The Celestial Intermediate Pole coordinates are the x,y
**     components of the unit vector in the Geocentric Celestial
**     Reference System.
**
**  3) The arguments xp and yp are the coordinates (in radians) of the
**     Celestial Intermediate Pole with respect to the International
**     Terrestrial Reference System (see IERS Conventions 2003),
**     measured along the meridians 0 and 90 deg west respectively.
**
**  4) The matrix rc2t transforms from celestial to terrestrial
**     coordinates:
**
**        [TRS] = RPOM * R_3(ERA) * RC2I * [CRS]
**
**              = rc2t * [CRS]
**
**     where [CRS] is a vector in the Geocentric Celestial Reference
**     System and [TRS] is a vector in the International Terrestrial
**     Reference System (see IERS Conventions 2003), ERA is the Earth
**     Rotation Angle and RPOM is the polar motion matrix.
**
**  5) Although its name does not include "00", This function is in fact
**     specific to the IAU 2000 models.
```

```
**
**  Called:
**     iauC2ixy     celestial-to-intermediate matrix, given X,Y
**     iauEra00     Earth rotation angle, IAU 2000
**     iauSp00      the TIO locator s', IERS 2000
**     iauPom00     polar motion matrix
**     iauC2tcio    form CIO-based celestial-to-terrestrial matrix
**
** Reference:
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG (2004)
**
*/
```

```
int iauCal2jd(int iy, int im, int id, double *djm0, double *djm)
/*
**  - - - - - - - - - -
**   i a u C a l 2 j d
**  - - - - - - - - - -
**
**  Gregorian Calendar to Julian Date.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     iy,im,id  int     year, month, day in Gregorian calendar (Note 1)
**
**  Returned:
**     djm0      double  MJD zero-point: always 2400000.5
**     djm       double  Modified Julian Date for 0 hrs
**
**  Returned (function value):
**               int     status:
**                           0 = OK
**                          -1 = bad year   (Note 3: JD not computed)
**                          -2 = bad month  (JD not computed)
**                          -3 = bad day    (JD computed)
**
**  Notes:
**
**  1) The algorithm used is valid from -4800 March 1, but this
**     implementation rejects dates before -4799 January 1.
**
**  2) The Julian Date is returned in two pieces, in the usual SOFA
**     manner, which is designed to preserve time resolution.  The
**     Julian Date is available as a single number by adding djm0 and
**     djm.
**
**  3) In early eras the conversion is from the "Proleptic Gregorian
**     Calendar";  no account is taken of the date(s) of adoption of
**     the Gregorian Calendar, nor is the AD/BC numbering convention
**     observed.
**
**  Reference:
**
**     Explanatory Supplement to the Astronomical Almanac,
**     P. Kenneth Seidelmann (ed), University Science Books (1992),
**     Section 12.92 (p604).
**
*/
```

```
void iauCp(double p[3], double c[3])
/*
**  - - - - - -
**   i a u C p
**  - - - - - -
**
**  Copy a p-vector.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     p        double[3]     p-vector to be copied
**
**  Returned:
**     c        double[3]     copy
**
*/
```

```
void iauCpv(double pv[2][3], double c[2][3])
/*
**  - - - - - - -
**   i a u C p v
**  - - - - - - -
**
**  Copy a position/velocity vector.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     pv     double[2][3]    position/velocity vector to be copied
**
**  Returned:
**     c      double[2][3]    copy
**
**  Called:
**     iauCp        copy p-vector
**
*/
```

```
void iauCr(double r[3][3], double c[3][3])
/*
**  - - - - - -
**   i a u C r
**  - - - - - -
**
**  Copy an r-matrix.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     r        double[3][3]    r-matrix to be copied
**
**  Returned:
**     c        double[3][3]    copy
**
**  Called:
**     iauCp        copy p-vector
**
*/
```

```
int iauD2dtf(const char *scale, int ndp, double d1, double d2,
             int *iy, int *im, int *id, int ihmsf[4])
/*
**  - - - - - - - - -
**   i a u D 2 d t f
**  - - - - - - - - -
**
**  Format for output a 2-part Julian Date (or in the case of UTC a
**  quasi-JD form that includes special provision for leap seconds).
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     scale     char[]  time scale ID (Note 1)
**     ndp       int     resolution (Note 2)
**     d1,d2     double  time as a 2-part Julian Date (Notes 3,4)
**
**  Returned:
**     iy,im,id  int     year, month, day in Gregorian calendar (Note 5)
**     ihmsf     int[4]  hours, minutes, seconds, fraction (Note 1)
**
**  Returned (function value):
**               int     status: +1 = dubious year (Note 5)
**                                0 = OK
**                               -1 = unacceptable date (Note 6)
**
**  Notes:
**
**  1) scale identifies the time scale.  Only the value "UTC" (in upper
**     case) is significant, and enables handling of leap seconds (see
**     Note 4).
**
**  2) ndp is the number of decimal places in the seconds field, and can
**     have negative as well as positive values, such as:
**
**     ndp         resolution
**     -4            1 00 00
**     -3            0 10 00
**     -2            0 01 00
**     -1            0 00 10
**      0            0 00 01
**      1            0 00 00.1
**      2            0 00 00.01
**      3            0 00 00.001
**
**     The limits are platform dependent, but a safe range is -5 to +9.
**
**  3) d1+d2 is Julian Date, apportioned in any convenient way between
**     the two arguments, for example where d1 is the Julian Day Number
**     and d2 is the fraction of a day.  In the case of UTC, where the
**     use of JD is problematical, special conventions apply:  see the
**     next note.
**
**  4) JD cannot unambiguously represent UTC during a leap second unless
**     special measures are taken.  The SOFA internal convention is that
**     the quasi-JD day represents UTC days whether the length is 86399,
**     86400 or 86401 SI seconds.  In the 1960-1972 era there were
**     smaller jumps (in either direction) each time the linear UTC(TAI)
**     expression was changed, and these "mini-leaps" are also included
**     in the SOFA convention.
**
**  5) The warning status "dubious year" flags UTCs that predate the
**     introduction of the time scale or that are too far in the future
**     to be trusted.  See iauDat for further details.
**
**  6) For calendar conventions and limitations, see iauCal2jd.
**
**  Called:
```

```
**     iauJd2cal    JD to Gregorian calendar
**     iauD2tf      decompose days to hms
**     iauDat       delta(AT) = TAI-UTC
**
*/
```

```
void iauD2tf(int ndp, double days, char *sign, int ihmsf[4])
/*
**  - - - - - - - -
**   i a u D 2 t f
**  - - - - - - - -
**
**  Decompose days to hours, minutes, seconds, fraction.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     ndp     int     resolution (Note 1)
**     days    double  interval in days
**
**  Returned:
**     sign    char*   '+' or '-'
**     ihmsf   int[4]  hours, minutes, seconds, fraction
**
**  Notes:
**
**  1) The argument ndp is interpreted as follows:
**
**     ndp         resolution
**      :      ...0000 00 00
**     -7         1000 00 00
**     -6          100 00 00
**     -5           10 00 00
**     -4            1 00 00
**     -3            0 10 00
**     -2            0 01 00
**     -1            0 00 10
**      0            0 00 01
**      1            0 00 00.1
**      2            0 00 00.01
**      3            0 00 00.001
**      :            0 00 00.000...
**
**  2) The largest positive useful value for ndp is determined by the
**     size of days, the format of double on the target platform, and
**     the risk of overflowing ihmsf[3].  On a typical platform, for
**     days up to 1.0, the available floating-point precision might
**     correspond to ndp=12.  However, the practical limit is typically
**     ndp=9, set by the capacity of a 32-bit int, or ndp=4 if int is
**     only 16 bits.
**
**  3) The absolute value of days may exceed 1.0.  In cases where it
**     does not, it is up to the caller to test for and handle the
**     case where days is very nearly 1.0 and rounds up to 24 hours,
**     by testing for ihmsf[0]=24 and setting ihmsf[0-3] to zero.
**
*/
```

```
int iauDat(int iy, int im, int id, double fd, double *deltat)
/*
**  - - - - - - - -
**   i a u D a t
**  - - - - - - - -
**
**  For a given UTC date, calculate Delta(AT) = TAI-UTC.
**
**     :------------------------------------------:
**     :                                          :
**     :                 IMPORTANT                :
**     :                                          :
**     :  A new version of this function must be  :
**     :  produced whenever a new leap second is  :
**     :  announced.  There are four items to     :
**     :  change on each such occasion:           :
**     :                                          :
**     :  1) A new line must be added to the set  :
**     :     of statements that initialize the    :
**     :     array "changes".                     :
**     :                                          :
**     :  2) The constant IYV must be set to the  :
**     :     current year.                        :
**     :                                          :
**     :  3) The "Latest leap second" comment     :
**     :     below must be set to the new leap    :
**     :     second date.                         :
**     :                                          :
**     :  4) The "This revision" comment, later,  :
**     :     must be set to the current date.     :
**     :                                          :
**     :  Change (2) must also be carried out     :
**     :  whenever the function is re-issued,     :
**     :  even if no leap seconds have been       :
**     :  added.                                  :
**     :                                          :
**     :  Latest leap second:  2016 December 31   :
**     :                                          :
**     :__________________________________________:
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  user-replaceable support function.
**
**  Given:
**     iy     int      UTC:  year (Notes 1 and 2)
**     im     int            month (Note 2)
**     id     int            day (Notes 2 and 3)
**     fd     double         fraction of day (Note 4)
**
**  Returned:
**     deltat double   TAI minus UTC, seconds
**
**  Returned (function value):
**            int      status (Note 5):
**                       1 = dubious year (Note 1)
**                       0 = OK
**                      -1 = bad year
**                      -2 = bad month
**                      -3 = bad day (Note 3)
**                      -4 = bad fraction (Note 4)
**                      -5 = internal error (Note 5)
**
**  Notes:
**
**  1) UTC began at 1960 January 1.0 (JD 2436934.5) and it is improper
**     to call the function with an earlier date.  If this is attempted,
**     zero is returned together with a warning status.
**
**     Because leap seconds cannot, in principle, be predicted in
```

```
**     advance, a reliable check for dates beyond the valid range is
**     impossible.  To guard against gross errors, a year five or more
**     after the release year of the present function (see the constant
**     IYV) is considered dubious.  In this case a warning status is
**     returned but the result is computed in the normal way.
**
**     For both too-early and too-late years, the warning status is +1.
**     This is distinct from the error status -1, which signifies a year
**     so early that JD could not be computed.
**
**  2) If the specified date is for a day which ends with a leap second,
**     the TAI-UTC value returned is for the period leading up to the
**     leap second.  If the date is for a day which begins as a leap
**     second ends, the TAI-UTC returned is for the period following the
**     leap second.
**
**  3) The day number must be in the normal calendar range, for example
**     1 through 30 for April.  The "almanac" convention of allowing
**     such dates as January 0 and December 32 is not supported in this
**     function, in order to avoid confusion near leap seconds.
**
**  4) The fraction of day is used only for dates before the
**     introduction of leap seconds, the first of which occurred at the
**     end of 1971.  It is tested for validity (0 to 1 is the valid
**     range) even if not used;  if invalid, zero is used and status -4
**     is returned.  For many applications, setting fd to zero is
**     acceptable;  the resulting error is always less than 3 ms (and
**     occurs only pre-1972).
**
**  5) The status value returned in the case where there are multiple
**     errors refers to the first error detected.  For example, if the
**     month and day are 13 and 32 respectively, status -2 (bad month)
**     will be returned.  The "internal error" status refers to a
**     case that is impossible but causes some compilers to issue a
**     warning.
**
**  6) In cases where a valid result is not available, zero is returned.
**
**  References:
**
**  1) For dates from 1961 January 1 onwards, the expressions from the
**     file ftp://maia.usno.navy.mil/ser7/tai-utc.dat are used.
**
**  2) The 5ms timestep at 1961 January 1 is taken from 2.58.1 (p87) of
**     the 1992 Explanatory Supplement.
**
**  Called:
**     iauCal2jd    Gregorian calendar to JD
**
*/
```

```
double iauDtdb(double date1, double date2,
               double ut, double elong, double u, double v)
/*
**  - - - - - - - -
**   i a u D t d b
**  - - - - - - - -
**
**  An approximation to TDB-TT, the difference between barycentric
**  dynamical time and terrestrial time, for an observer on the Earth.
**
**  The different time scales - proper, coordinate and realized - are
**  related to each other:
**
**            TAI             <-  physically realized
**             :
**          offset            <-  observed (nominally +32.184s)
**             :
**            TT              <-  terrestrial time
**             :
**    rate adjustment (L_G)   <-  definition of TT
**             :
**            TCG             <-  time scale for GCRS
**             :
**      "periodic" terms      <-  iauDtdb  is an implementation
**             :
**    rate adjustment (L_C)   <-  function of solar-system ephemeris
**             :
**            TCB             <-  time scale for BCRS
**             :
**    rate adjustment (-L_B)  <-  definition of TDB
**             :
**            TDB             <-  TCB scaled to track TT
**             :
**      "periodic" terms      <-  -iauDtdb is an approximation
**             :
**            TT              <-  terrestrial time
**
**  Adopted values for the various constants can be found in the IERS
**  Conventions (McCarthy & Petit 2003).
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1,date2   double  date, TDB (Notes 1-3)
**     ut            double  universal time (UT1, fraction of one day)
**     elong         double  longitude (east positive, radians)
**     u             double  distance from Earth spin axis (km)
**     v             double  distance north of equatorial plane (km)
**
**  Returned (function value):
**                   double  TDB-TT (seconds)
**
**  Notes:
**
**  1) The date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
```

```
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**     Although the date is, formally, barycentric dynamical time (TDB),
**     the terrestrial dynamical time (TT) can be used with no practical
**     effect on the accuracy of the prediction.
**
**  2) TT can be regarded as a coordinate time that is realized as an
**     offset of 32.184s from International Atomic Time, TAI.  TT is a
**     specific linear transformation of geocentric coordinate time TCG,
**     which is the time scale for the Geocentric Celestial Reference
**     System, GCRS.
**
**  3) TDB is a coordinate time, and is a specific linear transformation
**     of barycentric coordinate time TCB, which is the time scale for
**     the Barycentric Celestial Reference System, BCRS.
**
**  4) The difference TCG-TCB depends on the masses and positions of the
**     bodies of the solar system and the velocity of the Earth.  It is
**     dominated by a rate difference, the residual being of a periodic
**     character.  The latter, which is modeled by the present function,
**     comprises a main (annual) sinusoidal term of amplitude
**     approximately 0.00166 seconds, plus planetary terms up to about
**     20 microseconds, and lunar and diurnal terms up to 2 microseconds.
**     These effects come from the changing transverse Doppler effect
**     and gravitational red-shift as the observer (on the Earth's
**     surface) experiences variations in speed (with respect to the
**     BCRS) and gravitational potential.
**
**  5) TDB can be regarded as the same as TCB but with a rate adjustment
**     to keep it close to TT, which is convenient for many applications.
**     The history of successive attempts to define TDB is set out in
**     Resolution 3 adopted by the IAU General Assembly in 2006, which
**     defines a fixed TDB(TCB) transformation that is consistent with
**     contemporary solar-system ephemerides.  Future ephemerides will
**     imply slightly changed transformations between TCG and TCB, which
**     could introduce a linear drift between TDB and TT;  however, any
**     such drift is unlikely to exceed 1 nanosecond per century.
**
**  6) The geocentric TDB-TT model used in the present function is that of
**     Fairhead & Bretagnon (1990), in its full form.  It was originally
**     supplied by Fairhead (private communications with P.T.Wallace,
**     1990) as a Fortran subroutine.  The present C function contains an
**     adaptation of the Fairhead code.  The numerical results are
**     essentially unaffected by the changes, the differences with
**     respect to the Fairhead & Bretagnon original being at the 1e-20 s
**     level.
**
**     The topocentric part of the model is from Moyer (1981) and
**     Murray (1983), with fundamental arguments adapted from
**     Simon et al. 1994.  It is an approximation to the expression
**     ( v / c ) . ( r / c ), where v is the barycentric velocity of
**     the Earth, r is the geocentric position of the observer and
**     c is the speed of light.
**
**     By supplying zeroes for u and v, the topocentric part of the
**     model can be nullified, and the function will return the Fairhead
**     & Bretagnon result alone.
**
**  7) During the interval 1950-2050, the absolute accuracy is better
**     than +/- 3 nanoseconds relative to time ephemerides obtained by
**     direct numerical integrations based on the JPL DE405 solar system
**     ephemeris.
**
**  8) It must be stressed that the present function is merely a model,
**     and that numerical integration of solar-system ephemerides is the
**     definitive method for predicting the relationship between TCG and
**     TCB and hence between TT and TDB.
**
**  References:
**
```

```
**      Fairhead, L., & Bretagnon, P., Astron.Astrophys., 229, 240-247
**      (1990).
**
**      IAU 2006 Resolution 3.
**
**      McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**      IERS Technical Note No. 32, BKG (2004)
**
**      Moyer, T.D., Cel.Mech., 23, 33 (1981).
**
**      Murray, C.A., Vectorial Astrometry, Adam Hilger (1983).
**
**      Seidelmann, P.K. et al., Explanatory Supplement to the
**      Astronomical Almanac, Chapter 2, University Science Books (1992).
**
**      Simon, J.L., Bretagnon, P., Chapront, J., Chapront-Touze, M.,
**      Francou, G. & Laskar, J., Astron.Astrophys., 282, 663-683 (1994).
**
*/
```

```
int iauDtf2d(const char *scale, int iy, int im, int id,
             int ihr, int imn, double sec, double *d1, double *d2)
/*
**  - - - - - - - - - -
**   i a u D t f 2 d
**  - - - - - - - - - -
**
**  Encode date and time fields into 2-part Julian Date (or in the case
**  of UTC a quasi-JD form that includes special provision for leap
**  seconds).
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     scale     char[]  time scale ID (Note 1)
**     iy,im,id  int     year, month, day in Gregorian calendar (Note 2)
**     ihr,imn   int     hour, minute
**     sec       double  seconds
**
**  Returned:
**     d1,d2     double  2-part Julian Date (Notes 3,4)
**
**  Returned (function value):
**               int     status: +3 = both of next two
**                               +2 = time is after end of day (Note 5)
**                               +1 = dubious year (Note 6)
**                                0 = OK
**                               -1 = bad year
**                               -2 = bad month
**                               -3 = bad day
**                               -4 = bad hour
**                               -5 = bad minute
**                               -6 = bad second (<0)
**
**  Notes:
**
**  1) scale identifies the time scale.  Only the value "UTC" (in upper
**     case) is significant, and enables handling of leap seconds (see
**     Note 4).
**
**  2) For calendar conventions and limitations, see iauCal2jd.
**
**  3) The sum of the results, d1+d2, is Julian Date, where normally d1
**     is the Julian Day Number and d2 is the fraction of a day.  In the
**     case of UTC, where the use of JD is problematical, special
**     conventions apply:  see the next note.
**
**  4) JD cannot unambiguously represent UTC during a leap second unless
**     special measures are taken.  The SOFA internal convention is that
**     the quasi-JD day represents UTC days whether the length is 86399,
**     86400 or 86401 SI seconds.  In the 1960-1972 era there were
**     smaller jumps (in either direction) each time the linear UTC(TAI)
**     expression was changed, and these "mini-leaps" are also included
**     in the SOFA convention.
**
**  5) The warning status "time is after end of day" usually means that
**     the sec argument is greater than 60.0.  However, in a day ending
**     in a leap second the limit changes to 61.0 (or 59.0 in the case
**     of a negative leap second).
**
**  6) The warning status "dubious year" flags UTCs that predate the
**     introduction of the time scale or that are too far in the future
**     to be trusted.  See iauDat for further details.
**
**  7) Only in the case of continuous and regular time scales (TAI, TT,
**     TCG, TCB and TDB) is the result d1+d2 a Julian Date, strictly
**     speaking.  In the other cases (UT1 and UTC) the result must be
**     used with circumspection;  in particular the difference between
```

```
**     two such results cannot be interpreted as a precise time
**     interval.
**
**  Called:
**     iauCal2jd    Gregorian calendar to JD
**     iauDat       delta(AT) = TAI-UTC
**     iauJd2cal    JD to Gregorian calendar
**
*/
```

```
void iauEceq06(double date1, double date2, double dl, double db,
               double *dr, double *dd)
/*
**  - - - - - - - - - - -
**   i a u E c e q 0 6
**  - - - - - - - - - - -
**
**  Transformation from ecliptic coordinates (mean equinox and ecliptic
**  of date) to ICRS RA,Dec, using the IAU 2006 precession model.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1,date2 double TT as a 2-part Julian date (Note 1)
**     dl,db       double ecliptic longitude and latitude (radians)
**
**  Returned:
**     dr,dd       double ICRS right ascension and declination (radians)
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) No assumptions are made about whether the coordinates represent
**     starlight and embody astrometric effects such as parallax or
**     aberration.
**
**  3) The transformation is approximately that from ecliptic longitude
**     and latitude (mean equinox and ecliptic of date) to mean J2000.0
**     right ascension and declination, with only frame bias (always
**     less than 25 mas) to disturb this classical picture.
**
**  Called:
**     iauS2c       spherical coordinates to unit vector
**     iauEcm06     J2000.0 to ecliptic rotation matrix, IAU 2006
**     iauTrxp      product of transpose of r-matrix and p-vector
**     iauC2s       unit vector to spherical coordinates
**     iauAnp       normalize angle into range 0 to 2pi
**     iauAnpm      normalize angle into range +/- pi
**
*/
```

```
void iauEcm06(double date1, double date2, double rm[3][3])
/*
**  - - - - - - - - -
**   i a u E c m 0 6
**  - - - - - - - - -
**
**  ICRS equatorial to ecliptic rotation matrix, IAU 2006.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1,date2  double         TT as a 2-part Julian date (Note 1)
**
**  Returned:
**     rm           double[3][3]   ICRS to ecliptic rotation matrix
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The matrix is in the sense
**
**        E_ep = rm x P_ICRS,
**
**     where P_ICRS is a vector with respect to ICRS right ascension
**     and declination axes and E_ep is the same vector with respect to
**     the (inertial) ecliptic and equinox of date.
**
**     P_ICRS is a free vector, merely a direction, typically of unit
**     magnitude, and not bound to any particular spatial origin, such
**     as the Earth, Sun or SSB.  No assumptions are made about whether
**     it represents starlight and embodies astrometric effects such as
**     parallax or aberration.  The transformation is approximately that
**     between mean J2000.0 right ascension and declination and ecliptic
**     longitude and latitude, with only frame bias (always less than
**     25 mas) to disturb this classical picture.
**
**  Called:
**     iauObl06     mean obliquity, IAU 2006
**     iauPmat06    PB matrix, IAU 2006
**     iauIr        initialize r-matrix to identity
**     iauRx        rotate around X-axis
**     iauRxr       product of two r-matrices
**
*/
```

```
double iauEe00(double date1, double date2, double epsa, double dpsi)
/*
**  - - - - - - - -
**   i a u E e 0 0
**  - - - - - - - -
**
**  The equation of the equinoxes, compatible with IAU 2000 resolutions,
**  given the nutation in longitude and the mean obliquity.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical model.
**
**  Given:
**     date1,date2  double    TT as a 2-part Julian Date (Note 1)
**     epsa         double    mean obliquity (Note 2)
**     dpsi         double    nutation in longitude (Note 3)
**
**  Returned (function value):
**                  double    equation of the equinoxes (Note 4)
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The obliquity, in radians, is mean of date.
**
**  3) The result, which is in radians, operates in the following sense:
**
**        Greenwich apparent ST = GMST + equation of the equinoxes
**
**  4) The result is compatible with the IAU 2000 resolutions.  For
**     further details, see IERS Conventions 2003 and Capitaine et al.
**     (2002).
**
**  Called:
**     iauEect00    equation of the equinoxes complementary terms
**
**  References:
**
**     Capitaine, N., Wallace, P.T. and McCarthy, D.D., "Expressions to
**     implement the IAU 2000 definition of UT1", Astronomy &
**     Astrophysics, 406, 1135-1149 (2003)
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG (2004)
**
*/
```

```
double iauEe00a(double date1, double date2)
/*
**  - - - - - - - - - -
**   i a u E e 0 0 a
**  - - - - - - - - - -
**
**  Equation of the equinoxes, compatible with IAU 2000 resolutions.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1,date2  double    TT as a 2-part Julian Date (Note 1)
**
**  Returned (function value):
**                  double    equation of the equinoxes (Note 2)
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The result, which is in radians, operates in the following sense:
**
**        Greenwich apparent ST = GMST + equation of the equinoxes
**
**  3) The result is compatible with the IAU 2000 resolutions.  For
**     further details, see IERS Conventions 2003 and Capitaine et al.
**     (2002).
**
**  Called:
**     iauPr00      IAU 2000 precession adjustments
**     iauObl80     mean obliquity, IAU 1980
**     iauNut00a    nutation, IAU 2000A
**     iauEe00      equation of the equinoxes, IAU 2000
**
**  References:
**
**     Capitaine, N., Wallace, P.T. and McCarthy, D.D., "Expressions to
**     implement the IAU 2000 definition of UT1", Astronomy &
**     Astrophysics, 406, 1135-1149 (2003).
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG (2004).
**
*/
```

```
double iauEe00b(double date1, double date2)
/*
**  - - - - - - - - - -
**   i a u E e 0 0 b
**  - - - - - - - - - -
**
**  Equation of the equinoxes, compatible with IAU 2000 resolutions but
**  using the truncated nutation model IAU 2000B.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1,date2  double    TT as a 2-part Julian Date (Note 1)
**
**  Returned (function value):
**                  double    equation of the equinoxes (Note 2)
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The result, which is in radians, operates in the following sense:
**
**        Greenwich apparent ST = GMST + equation of the equinoxes
**
**  3) The result is compatible with the IAU 2000 resolutions except
**     that accuracy has been compromised (1 mas) for the sake of speed.
**     For further details, see McCarthy & Luzum (2003), IERS
**     Conventions 2003 and Capitaine et al. (2003).
**
**  Called:
**     iauPr00      IAU 2000 precession adjustments
**     iauObl80     mean obliquity, IAU 1980
**     iauNut00b    nutation, IAU 2000B
**     iauEe00      equation of the equinoxes, IAU 2000
**
**  References:
**
**     Capitaine, N., Wallace, P.T. and McCarthy, D.D., "Expressions to
**     implement the IAU 2000 definition of UT1", Astronomy &
**     Astrophysics, 406, 1135-1149 (2003)
**
**     McCarthy, D.D. & Luzum, B.J., "An abridged model of the
**     precession-nutation of the celestial pole", Celestial Mechanics &
**     Dynamical Astronomy, 85, 37-49 (2003)
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG (2004)
**
*/
```

```
double iauEe06a(double date1, double date2)
/*
**  - - - - - - - - -
**   i a u E e 0 6 a
**  - - - - - - - - -
**
**  Equation of the equinoxes, compatible with IAU 2000 resolutions and
**  IAU 2006/2000A precession-nutation.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1,date2  double    TT as a 2-part Julian Date (Note 1)
**
**  Returned (function value):
**                  double    equation of the equinoxes (Note 2)
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The result, which is in radians, operates in the following sense:
**
**        Greenwich apparent ST = GMST + equation of the equinoxes
**
**  Called:
**     iauAnpm      normalize angle into range +/- pi
**     iauGst06a    Greenwich apparent sidereal time, IAU 2006/2000A
**     iauGmst06    Greenwich mean sidereal time, IAU 2006
**
**  Reference:
**
**     McCarthy, D. D., Petit, G. (eds.), 2004, IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG
**
*/
```

```
double iauEect00(double date1, double date2)
/*
**  - - - - - - - - - -
**   i a u E e c t 0 0
**  - - - - - - - - - -
**
**  Equation of the equinoxes complementary terms, consistent with
**  IAU 2000 resolutions.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical model.
**
**  Given:
**     date1,date2  double   TT as a 2-part Julian Date (Note 1)
**
**  Returned (function value):
**                  double   complementary terms (Note 2)
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The "complementary terms" are part of the equation of the
**     equinoxes (EE), classically the difference between apparent and
**     mean Sidereal Time:
**
**        GAST = GMST + EE
**
**     with:
**
**        EE = dpsi * cos(eps)
**
**     where dpsi is the nutation in longitude and eps is the obliquity
**     of date.  However, if the rotation of the Earth were constant in
**     an inertial frame the classical formulation would lead to
**     apparent irregularities in the UT1 timescale traceable to side-
**     effects of precession-nutation.  In order to eliminate these
**     effects from UT1, "complementary terms" were introduced in 1994
**     (IAU, 1994) and took effect from 1997 (Capitaine and Gontier,
**     1993):
**
**        GAST = GMST + CT + EE
**
**     By convention, the complementary terms are included as part of
**     the equation of the equinoxes rather than as part of the mean
**     Sidereal Time.  This slightly compromises the "geometrical"
**     interpretation of mean sidereal time but is otherwise
**     inconsequential.
**
**     The present function computes CT in the above expression,
**     compatible with IAU 2000 resolutions (Capitaine et al., 2002, and
**     IERS Conventions 2003).
```

```
**
**  Called:
**     iauFal03     mean anomaly of the Moon
**     iauFalp03    mean anomaly of the Sun
**     iauFaf03     mean argument of the latitude of the Moon
**     iauFad03     mean elongation of the Moon from the Sun
**     iauFaom03    mean longitude of the Moon's ascending node
**     iauFave03    mean longitude of Venus
**     iauFae03     mean longitude of Earth
**     iauFapa03    general accumulated precession in longitude
**
**  References:
**
**     Capitaine, N. & Gontier, A.-M., Astron.Astrophys., 275,
**     645-650 (1993)
**
**     Capitaine, N., Wallace, P.T. and McCarthy, D.D., "Expressions to
**     implement the IAU 2000 definition of UT1", Astron.Astrophys., 406,
**     1135-1149 (2003)
**
**     IAU Resolution C7, Recommendation 3 (1994)
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG (2004)
**
*/
```

```
int iauEform ( int n, double *a, double *f )
/*
**  - - - - - - - - -
**   i a u E f o r m
**  - - - - - - - - -
**
**  Earth reference ellipsoids.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical.
**
**  Given:
**     n    int         ellipsoid identifier (Note 1)
**
**  Returned:
**     a    double      equatorial radius (meters, Note 2)
**     f    double      flattening (Note 2)
**
**  Returned (function value):
**          int         status:  0 = OK
**                              -1 = illegal identifier (Note 3)
**
**  Notes:
**
**  1) The identifier n is a number that specifies the choice of
**     reference ellipsoid.  The following are supported:
**
**        n    ellipsoid
**
**        1     WGS84
**        2     GRS80
**        3     WGS72
**
**     The n value has no significance outside the SOFA software.  For
**     convenience, symbols WGS84 etc. are defined in sofam.h.
**
**  2) The ellipsoid parameters are returned in the form of equatorial
**     radius in meters (a) and flattening (f).  The latter is a number
**     around 0.00335, i.e. around 1/298.
**
**  3) For the case where an unsupported n value is supplied, zero a and
**     f are returned, as well as error status.
**
**  References:
**
**     Department of Defense World Geodetic System 1984, National
**     Imagery and Mapping Agency Technical Report 8350.2, Third
**     Edition, p3-2.
**
**     Moritz, H., Bull. Geodesique 66-2, 187 (1992).
**
**     The Department of Defense World Geodetic System 1972, World
**     Geodetic System Committee, May 1974.
**
**     Explanatory Supplement to the Astronomical Almanac,
**     P. Kenneth Seidelmann (ed), University Science Books (1992),
**     p220.
**
*/
```

```
double iauEo06a(double date1, double date2)
/*
**  - - - - - - - - -
**   i a u E o 0 6 a
**  - - - - - - - - -
**
**  Equation of the origins, IAU 2006 precession and IAU 2000A nutation.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1,date2  double    TT as a 2-part Julian Date (Note 1)
**
**  Returned (function value):
**                  double    the equation of the origins in radians
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The equation of the origins is the distance between the true
**     equinox and the celestial intermediate origin and, equivalently,
**     the difference between Earth rotation angle and Greenwich
**     apparent sidereal time (ERA-GST).  It comprises the precession
**     (since J2000.0) in right ascension plus the equation of the
**     equinoxes (including the small correction terms).
**
**  Called:
**     iauPnm06a    classical NPB matrix, IAU 2006/2000A
**     iauBpn2xy    extract CIP X,Y coordinates from NPB matrix
**     iauS06       the CIO locator s, given X,Y, IAU 2006
**     iauEors      equation of the origins, given NPB matrix and s
**
**  References:
**
**     Capitaine, N. & Wallace, P.T., 2006, Astron.Astrophys. 450, 855
**
**     Wallace, P.T. & Capitaine, N., 2006, Astron.Astrophys. 459, 981
**
*/
```

```
double iauEors(double rnpb[3][3], double s)
/*
**  - - - - - - - -
**   i a u E o r s
**  - - - - - - - -
**
**  Equation of the origins, given the classical NPB matrix and the
**  quantity s.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     rnpb  double[3][3]  classical nutation x precession x bias matrix
**     s     double        the quantity s (the CIO locator) in radians
**
**  Returned (function value):
**           double        the equation of the origins in radians
**
**  Notes:
**
**  1)  The equation of the origins is the distance between the true
**      equinox and the celestial intermediate origin and, equivalently,
**      the difference between Earth rotation angle and Greenwich
**      apparent sidereal time (ERA-GST).  It comprises the precession
**      (since J2000.0) in right ascension plus the equation of the
**      equinoxes (including the small correction terms).
**
**  2)  The algorithm is from Wallace & Capitaine (2006).
**
** References:
**
**     Capitaine, N. & Wallace, P.T., 2006, Astron.Astrophys. 450, 855
**
**     Wallace, P. & Capitaine, N., 2006, Astron.Astrophys. 459, 981
**
*/
```

```
double iauEpb(double dj1, double dj2)
/*
**  - - - - - - - -
**   i a u E p b
**  - - - - - - - -
**
**  Julian Date to Besselian Epoch.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     dj1,dj2    double     Julian Date (Notes 3,4)
**
**  Returned (function value):
**                double     Besselian Epoch.
**
**  Notes:
**
**  1) Besselian Epoch is a method of expressing a moment in time as a
**     year plus fraction.  It was superseded by Julian Year (see the
**     function iauEpj).
**
**  2) The start of a Besselian year is when the right ascension of
**     the fictitious mean Sun is 18h 40m, and the unit is the tropical
**     year.  The conventional definition (see Lieske 1979) is that
**     Besselian Epoch B1900.0 is JD 2415020.31352 and the length of the
**     year is 365.242198781 days.
**
**  3) The time scale for the JD, originally Ephemeris Time, is TDB,
**     which for all practical purposes in the present context is
**     indistinguishable from TT.
**
**  4) The Julian Date is supplied in two pieces, in the usual SOFA
**     manner, which is designed to preserve time resolution.  The
**     Julian Date is available as a single number by adding dj1 and
**     dj2.  The maximum resolution is achieved if dj1 is 2451545.0
**     (J2000.0).
**
**  Reference:
**
**     Lieske, J.H., 1979. Astron.Astrophys., 73, 282.
**
*/
```

```
void iauEpb2jd(double epb, double *djm0, double *djm)
/*
**  - - - - - - - - - - -
**   i a u E p b 2 j d
**  - - - - - - - - - - -
**
**  Besselian Epoch to Julian Date.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     epb      double    Besselian Epoch (e.g. 1957.3)
**
**  Returned:
**     djm0     double    MJD zero-point: always 2400000.5
**     djm      double    Modified Julian Date
**
**  Note:
**
**     The Julian Date is returned in two pieces, in the usual SOFA
**     manner, which is designed to preserve time resolution.  The
**     Julian Date is available as a single number by adding djm0 and
**     djm.
**
**  Reference:
**
**     Lieske, J.H., 1979, Astron.Astrophys. 73, 282.
**
*/
```

```
double iauEpj(double dj1, double dj2)
/*
**  - - - - - - -
**   i a u E p j
**  - - - - - - -
**
**  Julian Date to Julian Epoch.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     dj1,dj2    double     Julian Date (Note 4)
**
**  Returned (function value):
**                double     Julian Epoch
**
**  Notes:
**
**  1) Julian Epoch is a method of expressing a moment in time as a
**     year plus fraction.
**
**  2) Julian Epoch J2000.0 is 2000 Jan 1.5, and the length of the year
**     is 365.25 days.
**
**  3) For historical reasons, the time scale formally associated with
**     Julian Epoch is TDB (or TT, near enough).  However, Julian Epoch
**     can be used more generally as a calendrical convention to
**     represent other time scales such as TAI and TCB.  This is
**     analogous to Julian Date, which was originally defined
**     specifically as a way of representing Universal Times but is now
**     routinely used for any of the regular time scales.
**
**  4) The Julian Date is supplied in two pieces, in the usual SOFA
**     manner, which is designed to preserve time resolution.  The
**     Julian Date is available as a single number by adding dj1 and
**     dj2.  The maximum resolution is achieved if dj1 is 2451545.0
**     (J2000.0).
**
**  Reference:
**
**     Lieske, J.H., 1979, Astron.Astrophys. 73, 282.
**
*/
```

```
void iauEpj2jd(double epj, double *djm0, double *djm)
/*
**  - - - - - - - - - -
**   i a u E p j 2 j d
**  - - - - - - - - - -
**
**  Julian Epoch to Julian Date.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     epj      double    Julian Epoch (e.g. 1996.8)
**
**  Returned:
**     djm0     double    MJD zero-point: always 2400000.5
**     djm      double    Modified Julian Date
**
**  Note:
**
**     The Julian Date is returned in two pieces, in the usual SOFA
**     manner, which is designed to preserve time resolution.  The
**     Julian Date is available as a single number by adding djm0 and
**     djm.
**
**  Reference:
**
**     Lieske, J.H., 1979, Astron.Astrophys. 73, 282.
**
*/
```

```
int iauEpv00(double date1, double date2,
             double pvh[2][3], double pvb[2][3])
/*
**  - - - - - - - - -
**   i a u E p v 0 0
**  - - - - - - - - -
**
**  Earth position and velocity, heliocentric and barycentric, with
**  respect to the Barycentric Celestial Reference System.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1,date2  double        TDB date (Note 1)
**
**  Returned:
**     pvh          double[2][3]  heliocentric Earth position/velocity
**     pvb          double[2][3]  barycentric Earth position/velocity
**
**  Returned (function value):
**                  int           status: 0 = OK
**                                       +1 = warning: date outside
**                                            the range 1900-2100 AD
**
**  Notes:
**
**  1) The TDB date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TDB)=2450123.7 could be expressed in any of these ways, among
**     others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in cases
**     where the loss of several decimal digits of resolution is
**     acceptable.  The J2000 method is best matched to the way the
**     argument is handled internally and will deliver the optimum
**     resolution.  The MJD method and the date & time methods are both
**     good compromises between resolution and convenience.  However,
**     the accuracy of the result is more likely to be limited by the
**     algorithm itself than the way the date has been expressed.
**
**     n.b. TT can be used instead of TDB in most applications.
**
**  2) On return, the arrays pvh and pvb contain the following:
**
**        pvh[0][0]  x       }
**        pvh[0][1]  y       } heliocentric position, au
**        pvh[0][2]  z       }
**
**        pvh[1][0]  xdot    }
**        pvh[1][1]  ydot    } heliocentric velocity, au/d
**        pvh[1][2]  zdot    }
**
**        pvb[0][0]  x       }
**        pvb[0][1]  y       } barycentric position, au
**        pvb[0][2]  z       }
**
**        pvb[1][0]  xdot    }
**        pvb[1][1]  ydot    } barycentric velocity, au/d
**        pvb[1][2]  zdot    }
**
**     The vectors are oriented with respect to the BCRS.  The time unit
```

```
**     is one day in TDB.
**
**  3) The function is a SIMPLIFIED SOLUTION from the planetary theory
**     VSOP2000 (X. Moisson, P. Bretagnon, 2001, Celes. Mechanics &
**     Dyn. Astron., 80, 3/4, 205-213) and is an adaptation of original
**     Fortran code supplied by P. Bretagnon (private comm., 2000).
**
**  4) Comparisons over the time span 1900-2100 with this simplified
**     solution and the JPL DE405 ephemeris give the following results:
**
**                                RMS    max
**           Heliocentric:
**              position error    3.7   11.2   km
**              velocity error    1.4    5.0   mm/s
**
**           Barycentric:
**              position error    4.6   13.4   km
**              velocity error    1.4    4.9   mm/s
**
**     Comparisons with the JPL DE406 ephemeris show that by 1800 and
**     2200 the position errors are approximately double their 1900-2100
**     size.  By 1500 and 2500 the deterioration is a factor of 10 and
**     by 1000 and 3000 a factor of 60.  The velocity accuracy falls off
**     at about half that rate.
**
**  5) It is permissible to use the same array for pvh and pvb, which
**     will receive the barycentric values.
**
*/
```

```
void iauEqec06(double date1, double date2, double dr, double dd,
               double *dl, double *db)
/*
**  - - - - - - - - - -
**   i a u E q e c 0 6
**  - - - - - - - - - -
**
**  Transformation from ICRS equatorial coordinates to ecliptic
**  coordinates (mean equinox and ecliptic of date) using IAU 2006
**  precession model.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1,date2 double TT as a 2-part Julian date (Note 1)
**     dr,dd       double ICRS right ascension and declination (radians)
**
**  Returned:
**     dl,db       double ecliptic longitude and latitude (radians)
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) No assumptions are made about whether the coordinates represent
**     starlight and embody astrometric effects such as parallax or
**     aberration.
**
**  3) The transformation is approximately that from mean J2000.0 right
**     ascension and declination to ecliptic longitude and latitude
**     (mean equinox and ecliptic of date), with only frame bias (always
**     less than 25 mas) to disturb this classical picture.
**
**  Called:
**     iauS2c       spherical coordinates to unit vector
**     iauEcm06     J2000.0 to ecliptic rotation matrix, IAU 2006
**     iauRxp       product of r-matrix and p-vector
**     iauC2s       unit vector to spherical coordinates
**     iauAnp       normalize angle into range 0 to 2pi
**     iauAnpm      normalize angle into range +/- pi
**
*/
```

```
double iauEqeq94(double date1, double date2)
/*
**  - - - - - - - - - -
**   i a u E q e q 9 4
**  - - - - - - - - - -
**
**  Equation of the equinoxes, IAU 1994 model.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical model.
**
**  Given:
**     date1,date2   double     TDB date (Note 1)
**
**  Returned (function value):
**                   double     equation of the equinoxes (Note 2)
**
**  Notes:
**
**  1) The date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The result, which is in radians, operates in the following sense:
**
**        Greenwich apparent ST = GMST + equation of the equinoxes
**
**  Called:
**     iauAnpm      normalize angle into range +/- pi
**     iauNut80     nutation, IAU 1980
**     iauObl80     mean obliquity, IAU 1980
**
**  References:
**
**     IAU Resolution C7, Recommendation 3 (1994).
**
**     Capitaine, N. & Gontier, A.-M., 1993, Astron.Astrophys., 275,
**     645-650.
**
*/
```

```
double iauEra00(double dj1, double dj2)
/*
**  - - - - - - - - -
**   i a u E r a 0 0
**  - - - - - - - - -
**
**  Earth rotation angle (IAU 2000 model).
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical model.
**
**  Given:
**     dj1,dj2   double    UT1 as a 2-part Julian Date (see note)
**
**  Returned (function value):
**               double    Earth rotation angle (radians), range 0-2pi
**
**  Notes:
**
**  1) The UT1 date dj1+dj2 is a Julian Date, apportioned in any
**     convenient way between the arguments dj1 and dj2.  For example,
**     JD(UT1)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**             dj1            dj2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 and MJD methods are good compromises
**     between resolution and convenience.  The date & time method is
**     best matched to the algorithm used:  maximum precision is
**     delivered when the dj1 argument is for 0hrs UT1 on the day in
**     question and the dj2 argument lies in the range 0 to 1, or vice
**     versa.
**
**  2) The algorithm is adapted from Expression 22 of Capitaine et al.
**     2000.  The time argument has been expressed in days directly,
**     and, to retain precision, integer contributions have been
**     eliminated.  The same formulation is given in IERS Conventions
**     (2003), Chap. 5, Eq. 14.
**
**  Called:
**     iauAnp       normalize angle into range 0 to 2pi
**
**  References:
**
**     Capitaine N., Guinot B. and McCarthy D.D, 2000, Astron.
**     Astrophys., 355, 398-405.
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG (2004)
**
*/
```

```
double iauFad03(double t)
/*
**  - - - - - - - - - -
**   i a u F a d 0 3
**  - - - - - - - - - -
**
**  Fundamental argument, IERS Conventions (2003):
**  mean elongation of the Moon from the Sun.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical model.
**
**  Given:
**     t     double    TDB, Julian centuries since J2000.0 (Note 1)
**
**  Returned (function value):
**           double    D, radians (Note 2)
**
**  Notes:
**
**  1) Though t is strictly TDB, it is usually more convenient to use
**     TT, which makes no significant difference.
**
**  2) The expression used is as adopted in IERS Conventions (2003) and
**     is from Simon et al. (1994).
**
**  References:
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG (2004)
**
**     Simon, J.-L., Bretagnon, P., Chapront, J., Chapront-Touze, M.,
**     Francou, G., Laskar, J. 1994, Astron.Astrophys. 282, 663-683
**
*/
```

```
double iauFae03(double t)
/*
**  - - - - - - - - - -
**   i a u F a e 0 3
**  - - - - - - - - - -
**
**  Fundamental argument, IERS Conventions (2003):
**  mean longitude of Earth.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical model.
**
**  Given:
**     t     double    TDB, Julian centuries since J2000.0 (Note 1)
**
**  Returned (function value):
**           double    mean longitude of Earth, radians (Note 2)
**
**  Notes:
**
**  1) Though t is strictly TDB, it is usually more convenient to use
**     TT, which makes no significant difference.
**
**  2) The expression used is as adopted in IERS Conventions (2003) and
**     comes from Souchay et al. (1999) after Simon et al. (1994).
**
**  References:
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG (2004)
**
**     Simon, J.-L., Bretagnon, P., Chapront, J., Chapront-Touze, M.,
**     Francou, G., Laskar, J. 1994, Astron.Astrophys. 282, 663-683
**
**     Souchay, J., Loysel, B., Kinoshita, H., Folgueira, M. 1999,
**     Astron.Astrophys.Supp.Ser. 135, 111
**
*/
```

```
double iauFaf03(double t)
/*
**  - - - - - - - - -
**   i a u F a f 0 3
**  - - - - - - - - -
**
**  Fundamental argument, IERS Conventions (2003):
**  mean longitude of the Moon minus mean longitude of the ascending
**  node.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical model.
**
**  Given:
**     t     double    TDB, Julian centuries since J2000.0 (Note 1)
**
**  Returned (function value):
**           double    F, radians (Note 2)
**
**  Notes:
**
**  1) Though t is strictly TDB, it is usually more convenient to use
**     TT, which makes no significant difference.
**
**  2) The expression used is as adopted in IERS Conventions (2003) and
**     is from Simon et al. (1994).
**
**  References:
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG (2004)
**
**     Simon, J.-L., Bretagnon, P., Chapront, J., Chapront-Touze, M.,
**     Francou, G., Laskar, J. 1994, Astron.Astrophys. 282, 663-683
**
*/
```

```
double iauFaju03(double t)
/*
**  - - - - - - - - - -
**   i a u F a j u 0 3
**  - - - - - - - - - -
**
**  Fundamental argument, IERS Conventions (2003):
**  mean longitude of Jupiter.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical model.
**
**  Given:
**     t     double    TDB, Julian centuries since J2000.0 (Note 1)
**
**  Returned (function value):
**           double    mean longitude of Jupiter, radians (Note 2)
**
**  Notes:
**
**  1) Though t is strictly TDB, it is usually more convenient to use
**     TT, which makes no significant difference.
**
**  2) The expression used is as adopted in IERS Conventions (2003) and
**     comes from Souchay et al. (1999) after Simon et al. (1994).
**
**  References:
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG (2004)
**
**     Simon, J.-L., Bretagnon, P., Chapront, J., Chapront-Touze, M.,
**     Francou, G., Laskar, J. 1994, Astron.Astrophys. 282, 663-683
**
**     Souchay, J., Loysel, B., Kinoshita, H., Folgueira, M. 1999,
**     Astron.Astrophys.Supp.Ser. 135, 111
**
*/
```

```
double iauFal03(double t)
/*
**  - - - - - - - - -
**   i a u F a l 0 3
**  - - - - - - - - -
**
**  Fundamental argument, IERS Conventions (2003):
**  mean anomaly of the Moon.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical model.
**
**  Given:
**     t     double    TDB, Julian centuries since J2000.0 (Note 1)
**
**  Returned (function value):
**           double    l, radians (Note 2)
**
**  Notes:
**
**  1) Though t is strictly TDB, it is usually more convenient to use
**     TT, which makes no significant difference.
**
**  2) The expression used is as adopted in IERS Conventions (2003) and
**     is from Simon et al. (1994).
**
**  References:
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG (2004)
**
**     Simon, J.-L., Bretagnon, P., Chapront, J., Chapront-Touze, M.,
**     Francou, G., Laskar, J. 1994, Astron.Astrophys. 282, 663-683
**
*/
```

```
double iauFalp03(double t)
/*
**  - - - - - - - - - - -
**   i a u F a l p 0 3
**  - - - - - - - - - - -
**
**  Fundamental argument, IERS Conventions (2003):
**  mean anomaly of the Sun.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical model.
**
**  Given:
**     t     double    TDB, Julian centuries since J2000.0 (Note 1)
**
**  Returned (function value):
**           double    l', radians (Note 2)
**
**  Notes:
**
**  1) Though t is strictly TDB, it is usually more convenient to use
**     TT, which makes no significant difference.
**
**  2) The expression used is as adopted in IERS Conventions (2003) and
**     is from Simon et al. (1994).
**
**  References:
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG (2004)
**
**     Simon, J.-L., Bretagnon, P., Chapront, J., Chapront-Touze, M.,
**     Francou, G., Laskar, J. 1994, Astron.Astrophys. 282, 663-683
**
*/
```

```
double iauFama03(double t)
/*
**  - - - - - - - - - -
**   i a u F a m a 0 3
**  - - - - - - - - - -
**
**  Fundamental argument, IERS Conventions (2003):
**  mean longitude of Mars.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical model.
**
**  Given:
**     t     double    TDB, Julian centuries since J2000.0 (Note 1)
**
**  Returned (function value):
**           double    mean longitude of Mars, radians (Note 2)
**
**  Notes:
**
**  1) Though t is strictly TDB, it is usually more convenient to use
**     TT, which makes no significant difference.
**
**  2) The expression used is as adopted in IERS Conventions (2003) and
**     comes from Souchay et al. (1999) after Simon et al. (1994).
**
**  References:
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG (2004)
**
**     Simon, J.-L., Bretagnon, P., Chapront, J., Chapront-Touze, M.,
**     Francou, G., Laskar, J. 1994, Astron.Astrophys. 282, 663-683
**
**     Souchay, J., Loysel, B., Kinoshita, H., Folgueira, M. 1999,
**     Astron.Astrophys.Supp.Ser. 135, 111
**
*/
```

```
double iauFame03(double t)
/*
**  - - - - - - - - - -
**   i a u F a m e 0 3
**  - - - - - - - - - -
**
**  Fundamental argument, IERS Conventions (2003):
**  mean longitude of Mercury.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical model.
**
**  Given:
**     t     double    TDB, Julian centuries since J2000.0 (Note 1)
**
**  Returned (function value):
**           double    mean longitude of Mercury, radians (Note 2)
**
**  Notes:
**
**  1) Though t is strictly TDB, it is usually more convenient to use
**     TT, which makes no significant difference.
**
**  2) The expression used is as adopted in IERS Conventions (2003) and
**     comes from Souchay et al. (1999) after Simon et al. (1994).
**
**  References:
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG (2004)
**
**     Simon, J.-L., Bretagnon, P., Chapront, J., Chapront-Touze, M.,
**     Francou, G., Laskar, J. 1994, Astron.Astrophys. 282, 663-683
**
**     Souchay, J., Loysel, B., Kinoshita, H., Folgueira, M. 1999,
**     Astron.Astrophys.Supp.Ser. 135, 111
**
*/
```

```
double iauFane03(double t)
/*
**  - - - - - - - - - -
**   i a u F a n e 0 3
**  - - - - - - - - - -
**
**  Fundamental argument, IERS Conventions (2003):
**  mean longitude of Neptune.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical model.
**
**  Given:
**     t     double    TDB, Julian centuries since J2000.0 (Note 1)
**
**  Returned (function value):
**           double    mean longitude of Neptune, radians (Note 2)
**
**  Notes:
**
**  1) Though t is strictly TDB, it is usually more convenient to use
**     TT, which makes no significant difference.
**
**  2) The expression used is as adopted in IERS Conventions (2003) and
**     is adapted from Simon et al. (1994).
**
**  References:
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG (2004)
**
**     Simon, J.-L., Bretagnon, P., Chapront, J., Chapront-Touze, M.,
**     Francou, G., Laskar, J. 1994, Astron.Astrophys. 282, 663-683
**
*/
```

```
double iauFaom03(double t)
/*
**  - - - - - - - - - -
**   i a u F a o m 0 3
**  - - - - - - - - - -
**
**  Fundamental argument, IERS Conventions (2003):
**  mean longitude of the Moon's ascending node.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical model.
**
**  Given:
**     t     double    TDB, Julian centuries since J2000.0 (Note 1)
**
**  Returned (function value):
**           double    Omega, radians (Note 2)
**
**  Notes:
**
**  1) Though t is strictly TDB, it is usually more convenient to use
**     TT, which makes no significant difference.
**
**  2) The expression used is as adopted in IERS Conventions (2003) and
**     is from Simon et al. (1994).
**
**  References:
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG (2004)
**
**     Simon, J.-L., Bretagnon, P., Chapront, J., Chapront-Touze, M.,
**     Francou, G., Laskar, J., 1994, Astron.Astrophys. 282, 663-683.
**
*/
```

```
double iauFapa03(double t)
/*
**  - - - - - - - - - - -
**   i a u F a p a 0 3
**  - - - - - - - - - - -
**
**  Fundamental argument, IERS Conventions (2003):
**  general accumulated precession in longitude.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical model.
**
**  Given:
**     t     double    TDB, Julian centuries since J2000.0 (Note 1)
**
**  Returned (function value):
**           double    general precession in longitude, radians (Note 2)
**
**  Notes:
**
**  1) Though t is strictly TDB, it is usually more convenient to use
**     TT, which makes no significant difference.
**
**  2) The expression used is as adopted in IERS Conventions (2003).  It
**     is taken from Kinoshita & Souchay (1990) and comes originally
**     from Lieske et al. (1977).
**
**  References:
**
**     Kinoshita, H. and Souchay J. 1990, Celest.Mech. and Dyn.Astron.
**     48, 187
**
**     Lieske, J.H., Lederle, T., Fricke, W. & Morando, B. 1977,
**     Astron.Astrophys. 58, 1-16
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG (2004)
**
*/
```

```
double iauFasa03(double t)
/*
**  - - - - - - - - - -
**   i a u F a s a 0 3
**  - - - - - - - - - -
**
**  Fundamental argument, IERS Conventions (2003):
**  mean longitude of Saturn.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical model.
**
**  Given:
**     t     double    TDB, Julian centuries since J2000.0 (Note 1)
**
**  Returned (function value):
**           double    mean longitude of Saturn, radians (Note 2)
**
**  Notes:
**
**  1) Though t is strictly TDB, it is usually more convenient to use
**     TT, which makes no significant difference.
**
**  2) The expression used is as adopted in IERS Conventions (2003) and
**     comes from Souchay et al. (1999) after Simon et al. (1994).
**
**  References:
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG (2004)
**
**     Simon, J.-L., Bretagnon, P., Chapront, J., Chapront-Touze, M.,
**     Francou, G., Laskar, J. 1994, Astron.Astrophys. 282, 663-683
**
**     Souchay, J., Loysel, B., Kinoshita, H., Folgueira, M. 1999,
**     Astron.Astrophys.Supp.Ser. 135, 111
**
*/
```

```
double iauFaur03(double t)
/*
**  - - - - - - - - - -
**   i a u F a u r 0 3
**  - - - - - - - - - -
**
**  Fundamental argument, IERS Conventions (2003):
**  mean longitude of Uranus.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical model.
**
**  Given:
**     t     double    TDB, Julian centuries since J2000.0 (Note 1)
**
**  Returned  (function value):
**           double    mean longitude of Uranus, radians (Note 2)
**
**  Notes:
**
**  1) Though t is strictly TDB, it is usually more convenient to use
**     TT, which makes no significant difference.
**
**  2) The expression used is as adopted in IERS Conventions (2003) and
**     is adapted from Simon et al. (1994).
**
**  References:
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG (2004)
**
**     Simon, J.-L., Bretagnon, P., Chapront, J., Chapront-Touze, M.,
**     Francou, G., Laskar, J. 1994, Astron.Astrophys. 282, 663-683
**
*/
```

```
double iauFave03(double t)
/*
**  - - - - - - - - - -
**   i a u F a v e 0 3
**  - - - - - - - - - -
**
**  Fundamental argument, IERS Conventions (2003):
**  mean longitude of Venus.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical model.
**
**  Given:
**     t     double    TDB, Julian centuries since J2000.0 (Note 1)
**
**  Returned (function value):
**           double    mean longitude of Venus, radians (Note 2)
**
**  Notes:
**
**  1) Though t is strictly TDB, it is usually more convenient to use
**     TT, which makes no significant difference.
**
**  2) The expression used is as adopted in IERS Conventions (2003) and
**     comes from Souchay et al. (1999) after Simon et al. (1994).
**
**  References:
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG (2004)
**
**     Simon, J.-L., Bretagnon, P., Chapront, J., Chapront-Touze, M.,
**     Francou, G., Laskar, J. 1994, Astron.Astrophys. 282, 663-683
**
**     Souchay, J., Loysel, B., Kinoshita, H., Folgueira, M. 1999,
**     Astron.Astrophys.Supp.Ser. 135, 111
**
*/
```

```
void iauFk425(double r1950, double d1950,
              double dr1950, double dd1950,
              double p1950, double v1950,
              double *r2000, double *d2000,
              double *dr2000, double *dd2000,
              double *p2000, double *v2000)
/*
**  - - - - - - - - -
**   i a u F k 4 2 5
**  - - - - - - - - -
**
**  Convert B1950.0 FK4 star catalog data to J2000.0 FK5.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  This function converts a star's catalog data from the old FK4
**  (Bessel-Newcomb) system to the later IAU 1976 FK5 (Fricke) system.
**
**  Given: (all B1950.0, FK4)
**     r1950,d1950    double   B1950.0 RA,Dec (rad)
**     dr1950,dd1950  double   B1950.0 proper motions (rad/trop.yr)
**     p1950          double   parallax (arcsec)
**     v1950          double   radial velocity (km/s, +ve = moving away)
**
**  Returned: (all J2000.0, FK5)
**     r2000,d2000    double   J2000.0 RA,Dec (rad)
**     dr2000,dd2000  double   J2000.0 proper motions (rad/Jul.yr)
**     p2000          double   parallax (arcsec)
**     v2000          double   radial velocity (km/s, +ve = moving away)
**
**  Notes:
**
**  1) The proper motions in RA are dRA/dt rather than cos(Dec)*dRA/dt,
**     and are per year rather than per century.
**
**  2) The conversion is somewhat complicated, for several reasons:
**
**     . Change of standard epoch from B1950.0 to J2000.0.
**
**     . An intermediate transition date of 1984 January 1.0 TT.
**
**     . A change of precession model.
**
**     . Change of time unit for proper motion (tropical to Julian).
**
**     . FK4 positions include the E-terms of aberration, to simplify
**       the hand computation of annual aberration.  FK5 positions
**       assume a rigorous aberration computation based on the Earth's
**       barycentric velocity.
**
**     . The E-terms also affect proper motions, and in particular cause
**       objects at large distances to exhibit fictitious proper
**       motions.
**
**     The algorithm is based on Smith et al. (1989) and Yallop et al.
**     (1989), which presented a matrix method due to Standish (1982) as
**     developed by Aoki et al. (1983), using Kinoshita's development of
**     Andoyer's post-Newcomb precession.  The numerical constants from
**     Seidelmann (1992) are used canonically.
**
**  3) Conversion from B1950.0 FK4 to J2000.0 FK5 only is provided for.
**     Conversions for different epochs and equinoxes would require
**     additional treatment for precession, proper motion and E-terms.
**
**  4) In the FK4 catalog the proper motions of stars within 10 degrees
**     of the poles do not embody differential E-terms effects and
**     should, strictly speaking, be handled in a different manner from
**     stars outside these regions.  However, given the general lack of
```

```
**     homogeneity of the star data available for routine astrometry,
**     the difficulties of handling positions that may have been
**     determined from astrometric fields spanning the polar and non-
**     polar regions, the likelihood that the differential E-terms
**     effect was not taken into account when allowing for proper motion
**     in past astrometry, and the undesirability of a discontinuity in
**     the algorithm, the decision has been made in this SOFA algorithm
**     to include the effects of differential E-terms on the proper
**     motions for all stars, whether polar or not.  At epoch J2000.0,
**     and measuring "on the sky" rather than in terms of RA change, the
**     errors resulting from this simplification are less than
**     1 milliarcsecond in position and 1 milliarcsecond per century in
**     proper motion.
**
**  Called:
**     iauAnp       normalize angle into range 0 to 2pi
**     iauPv2s      pv-vector to spherical coordinates
**     iauPdp       scalar product of two p-vectors
**     iauPvmpv     pv-vector minus pv_vector
**     iauPvppv     pv-vector plus pv_vector
**     iauS2pv      spherical coordinates to pv-vector
**     iauSxp       multiply p-vector by scalar
**
**  References:
**
**     Aoki, S. et al., 1983, "Conversion matrix of epoch B1950.0
**     FK4-based positions of stars to epoch J2000.0 positions in
**     accordance with the new IAU resolutions".  Astron.Astrophys.
**     128, 263-267.
**
**     Seidelmann, P.K. (ed), 1992, "Explanatory Supplement to the
**     Astronomical Almanac", ISBN 0-935702-68-7.
**
**     Smith, C.A. et al., 1989, "The transformation of astrometric
**     catalog systems to the equinox J2000.0".  Astron.J. 97, 265.
**
**     Standish, E.M., 1982, "Conversion of positions and proper motions
**     from B1950.0 to the IAU system at J2000.0".  Astron.Astrophys.,
**     115, 1, 20-22.
**
**     Yallop, B.D. et al., 1989, "Transformation of mean star places
**     from FK4 B1950.0 to FK5 J2000.0 using matrices in 6-space".
**     Astron.J. 97, 274.
**
*/
```

```
void iauFk45z(double r1950, double d1950, double bepoch,
              double *r2000, double *d2000)
/*
**  - - - - - - - - -
**   i a u F k 4 5 z
**  - - - - - - - - -
**
**  Convert a B1950.0 FK4 star position to J2000.0 FK5, assuming zero
**  proper motion in the FK5 system.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  This function converts a star's catalog data from the old FK4
**  (Bessel-Newcomb) system to the later IAU 1976 FK5 (Fricke) system,
**  in such a way that the FK5 proper motion is zero.  Because such a
**  star has, in general, a non-zero proper motion in the FK4 system,
**  the function requires the epoch at which the position in the FK4
**  system was determined.
**
**  Given:
**     r1950,d1950    double   B1950.0 FK4 RA,Dec at epoch (rad)
**     bepoch         double   Besselian epoch (e.g. 1979.3)
**
**  Returned:
**     r2000,d2000    double   J2000.0 FK5 RA,Dec (rad)
**
**  Notes:
**
**  1) The epoch bepoch is strictly speaking Besselian, but if a
**     Julian epoch is supplied the result will be affected only to a
**     negligible extent.
**
**  2) The method is from Appendix 2 of Aoki et al. (1983), but using
**     the constants of Seidelmann (1992).  See the function iauFk425
**     for a general introduction to the FK4 to FK5 conversion.
**
**  3) Conversion from equinox B1950.0 FK4 to equinox J2000.0 FK5 only
**     is provided for.  Conversions for different starting and/or
**     ending epochs would require additional treatment for precession,
**     proper motion and E-terms.
**
**  4) In the FK4 catalog the proper motions of stars within 10 degrees
**     of the poles do not embody differential E-terms effects and
**     should, strictly speaking, be handled in a different manner from
**     stars outside these regions.  However, given the general lack of
**     homogeneity of the star data available for routine astrometry,
**     the difficulties of handling positions that may have been
**     determined from astrometric fields spanning the polar and non-
**     polar regions, the likelihood that the differential E-terms
**     effect was not taken into account when allowing for proper motion
**     in past astrometry, and the undesirability of a discontinuity in
**     the algorithm, the decision has been made in this SOFA algorithm
**     to include the effects of differential E-terms on the proper
**     motions for all stars, whether polar or not.  At epoch J2000.0,
**     and measuring "on the sky" rather than in terms of RA change, the
**     errors resulting from this simplification are less than
**     1 milliarcsecond in position and 1 milliarcsecond per century in
**     proper motion.
**
**  References:
**
**     Aoki, S. et al., 1983, "Conversion matrix of epoch B1950.0
**     FK4-based positions of stars to epoch J2000.0 positions in
**     accordance with the new IAU resolutions".  Astron.Astrophys.
**     128, 263-267.
**
**     Seidelmann, P.K. (ed), 1992, "Explanatory Supplement to the
**     Astronomical Almanac", ISBN 0-935702-68-7.
```

```
**
**  Called:
**     iauAnp       normalize angle into range 0 to 2pi
**     iauC2s       p-vector to spherical
**     iauEpb2jd    Besselian epoch to Julian date
**     iauEpj       Julian date to Julian epoch
**     iauPdp       scalar product of two p-vectors
**     iauPmp       p-vector minus p-vector
**     iauPpsp      p-vector plus scaled p-vector
**     iauPvu       update a pv-vector
**     iauS2c       spherical to p-vector
**
*/
```

```
void iauFk524(double r2000, double d2000,
              double dr2000, double dd2000,
              double p2000, double v2000,
              double *r1950, double *d1950,
              double *dr1950, double *dd1950,
              double *p1950, double *v1950)
/*
**  - - - - - - - - - -
**   i a u F k 5 2 4
**  - - - - - - - - - -
**
**  Convert J2000.0 FK5 star catalog data to B1950.0 FK4.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given: (all J2000.0, FK5)
**     r2000,d2000    double   J2000.0 RA,Dec (rad)
**     dr2000,dd2000  double   J2000.0 proper motions (rad/Jul.yr)
**     p2000          double   parallax (arcsec)
**     v2000          double   radial velocity (km/s, +ve = moving away)
**
**  Returned: (all B1950.0, FK4)
**     r1950,d1950    double   B1950.0 RA,Dec (rad)
**     dr1950,dd1950  double   B1950.0 proper motions (rad/trop.yr)
**     p1950          double   parallax (arcsec)
**     v1950          double   radial velocity (km/s, +ve = moving away)
**
**  Notes:
**
**  1) The proper motions in RA are dRA/dt rather than cos(Dec)*dRA/dt,
**     and are per year rather than per century.
**
**  2) The conversion is somewhat complicated, for several reasons:
**
**     . Change of standard epoch from J2000.0 to B1950.0.
**
**     . An intermediate transition date of 1984 January 1.0 TT.
**
**     . A change of precession model.
**
**     . Change of time unit for proper motion (Julian to tropical).
**
**     . FK4 positions include the E-terms of aberration, to simplify
**       the hand computation of annual aberration.  FK5 positions
**       assume a rigorous aberration computation based on the Earth's
**       barycentric velocity.
**
**     . The E-terms also affect proper motions, and in particular cause
**       objects at large distances to exhibit fictitious proper
**       motions.
**
**     The algorithm is based on Smith et al. (1989) and Yallop et al.
**     (1989), which presented a matrix method due to Standish (1982) as
**     developed by Aoki et al. (1983), using Kinoshita's development of
**     Andoyer's post-Newcomb precession.  The numerical constants from
**     Seidelmann (1992) are used canonically.
**
**  4) In the FK4 catalog the proper motions of stars within 10 degrees
**     of the poles do not embody differential E-terms effects and
**     should, strictly speaking, be handled in a different manner from
**     stars outside these regions.  However, given the general lack of
**     homogeneity of the star data available for routine astrometry,
**     the difficulties of handling positions that may have been
**     determined from astrometric fields spanning the polar and non-
**     polar regions, the likelihood that the differential E-terms
**     effect was not taken into account when allowing for proper motion
**     in past astrometry, and the undesirability of a discontinuity in
**     the algorithm, the decision has been made in this SOFA algorithm
```

```
**     to include the effects of differential E-terms on the proper
**     motions for all stars, whether polar or not.  At epoch J2000.0,
**     and measuring "on the sky" rather than in terms of RA change, the
**     errors resulting from this simplification are less than
**     1 milliarcsecond in position and 1 milliarcsecond per century in
**     proper motion.
**
**  Called:
**     iauAnp       normalize angle into range 0 to 2pi
**     iauPdp       scalar product of two p-vectors
**     iauPm        modulus of p-vector
**     iauPmp       p-vector minus p-vector
**     iauPpp       p-vector pluus p-vector
**     iauPv2s      pv-vector to spherical coordinates
**     iauS2pv      spherical coordinates to pv-vector
**     iauSxp       multiply p-vector by scalar
**
**  References:
**
**     Aoki, S. et al., 1983, "Conversion matrix of epoch B1950.0
**     FK4-based positions of stars to epoch J2000.0 positions in
**     accordance with the new IAU resolutions".  Astron.Astrophys.
**     128, 263-267.
**
**     Seidelmann, P.K. (ed), 1992, "Explanatory Supplement to the
**     Astronomical Almanac", ISBN 0-935702-68-7.
**
**     Smith, C.A. et al., 1989, "The transformation of astrometric
**     catalog systems to the equinox J2000.0".  Astron.J. 97, 265.
**
**     Standish, E.M., 1982, "Conversion of positions and proper motions
**     from B1950.0 to the IAU system at J2000.0".  Astron.Astrophys.,
**     115, 1, 20-22.
**
**     Yallop, B.D. et al., 1989, "Transformation of mean star places
**     from FK4 B1950.0 to FK5 J2000.0 using matrices in 6-space".
**     Astron.J. 97, 274.
**
*/
```

```
void iauFk52h(double r5, double d5,
              double dr5, double dd5, double px5, double rv5,
              double *rh, double *dh,
              double *drh, double *ddh, double *pxh, double *rvh)
/*
**  - - - - - - - - -
**   i a u F k 5 2 h
**  - - - - - - - - -
**
**  Transform FK5 (J2000.0) star data into the Hipparcos system.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given (all FK5, equinox J2000.0, epoch J2000.0):
**     r5      double    RA (radians)
**     d5      double    Dec (radians)
**     dr5     double    proper motion in RA (dRA/dt, rad/Jyear)
**     dd5     double    proper motion in Dec (dDec/dt, rad/Jyear)
**     px5     double    parallax (arcsec)
**     rv5     double    radial velocity (km/s, positive = receding)
**
**  Returned (all Hipparcos, epoch J2000.0):
**     rh      double    RA (radians)
**     dh      double    Dec (radians)
**     drh     double    proper motion in RA (dRA/dt, rad/Jyear)
**     ddh     double    proper motion in Dec (dDec/dt, rad/Jyear)
**     pxh     double    parallax (arcsec)
**     rvh     double    radial velocity (km/s, positive = receding)
**
**  Notes:
**
**  1) This function transforms FK5 star positions and proper motions
**     into the system of the Hipparcos catalog.
**
**  2) The proper motions in RA are dRA/dt rather than
**     cos(Dec)*dRA/dt, and are per year rather than per century.
**
**  3) The FK5 to Hipparcos transformation is modeled as a pure
**     rotation and spin;  zonal errors in the FK5 catalog are not
**     taken into account.
**
**  4) See also iauH2fk5, iauFk5hz, iauHfk5z.
**
**  Called:
**     iauStarpv    star catalog data to space motion pv-vector
**     iauFk5hip    FK5 to Hipparcos rotation and spin
**     iauRxp       product of r-matrix and p-vector
**     iauPxp       vector product of two p-vectors
**     iauPpp       p-vector plus p-vector
**     iauPvstar    space motion pv-vector to star catalog data
**
**  Reference:
**
**     F.Mignard & M.Froeschle, Astron.Astrophys., 354, 732-739 (2000).
**
*/
```

```
void iauFk54z(double r2000, double d2000, double bepoch,
              double *r1950, double *d1950,
              double *dr1950, double *dd1950)
/*
**  - - - - - - - - -
**   i a u F k 5 4 z
**  - - - - - - - - -
**
**  Convert a J2000.0 FK5 star position to B1950.0 FK4, assuming zero
**  proper motion in FK5 and parallax.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     r2000,d2000    double   J2000.0 FK5 RA,Dec (rad)
**     bepoch         double   Besselian epoch (e.g. 1950.0)
**
**  Returned:
**     r1950,d1950    double   B1950.0 FK4 RA,Dec (rad) at epoch BEPOCH
**     dr1950,dd1950  double   B1950.0 FK4 proper motions (rad/trop.yr)
**
**  Notes:
**
**  1) In contrast to the iauFk524 function, here the FK5 proper
**     motions, the parallax and the radial velocity are presumed zero.
**
**  2) This function converts a star position from the IAU 1976 FK5
**    (Fricke) system to the former FK4 (Bessel-Newcomb) system, for
**     cases such as distant radio sources where it is presumed there is
**     zero parallax and no proper motion.  Because of the E-terms of
**     aberration, such objects have (in general) non-zero proper motion
**     in FK4, and the present function returns those fictitious proper
**     motions.
**
**  3) Conversion from J2000.0 FK5 to B1950.0 FK4 only is provided for.
**     Conversions involving other equinoxes would require additional
**     treatment for precession.
**
**  4) The position returned by this function is in the B1950.0 FK4
**     reference system but at Besselian epoch bepoch.  For comparison
**     with catalogs the bepoch argument will frequently be 1950.0. (In
**     this context the distinction between Besselian and Julian epoch
**     is insignificant.)
**
**  5) The RA component of the returned (fictitious) proper motion is
**     dRA/dt rather than cos(Dec)*dRA/dt.
**
**  Called:
**     iauAnp       normalize angle into range 0 to 2pi
**     iauC2s       p-vector to spherical
**     iauFk524     FK4 to FK5
**     iauS2c       spherical to p-vector
**
*/
```

```
void iauFk5hip(double r5h[3][3], double s5h[3])
/*
**  - - - - - - - - - -
**   i a u F k 5 h i p
**  - - - - - - - - - -
**
**  FK5 to Hipparcos rotation and spin.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Returned:
**     r5h   double[3][3]  r-matrix: FK5 rotation wrt Hipparcos (Note 2)
**     s5h   double[3]     r-vector: FK5 spin wrt Hipparcos (Note 3)
**
**  Notes:
**
**  1) This function models the FK5 to Hipparcos transformation as a
**     pure rotation and spin;  zonal errors in the FK5 catalog are not
**     taken into account.
**
**  2) The r-matrix r5h operates in the sense:
**
**           P_Hipparcos = r5h x P_FK5
**
**     where P_FK5 is a p-vector in the FK5 frame, and P_Hipparcos is
**     the equivalent Hipparcos p-vector.
**
**  3) The r-vector s5h represents the time derivative of the FK5 to
**     Hipparcos rotation.  The units are radians per year (Julian,
**     TDB).
**
**  Called:
**     iauRv2m      r-vector to r-matrix
**
**  Reference:
**
**     F.Mignard & M.Froeschle, Astron.Astrophys., 354, 732-739 (2000).
**
*/
```

```
void iauFk5hz(double r5, double d5, double date1, double date2,
              double *rh, double *dh)
/*
**  - - - - - - - - - -
**   i a u F k 5 h z
**  - - - - - - - - - -
**
**  Transform an FK5 (J2000.0) star position into the system of the
**  Hipparcos catalog, assuming zero Hipparcos proper motion.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     r5           double   FK5 RA (radians), equinox J2000.0, at date
**     d5           double   FK5 Dec (radians), equinox J2000.0, at date
**     date1,date2  double   TDB date (Notes 1,2)
**
**  Returned:
**     rh           double   Hipparcos RA (radians)
**     dh           double   Hipparcos Dec (radians)
**
**  Notes:
**
**  1) This function converts a star position from the FK5 system to
**     the Hipparcos system, in such a way that the Hipparcos proper
**     motion is zero.  Because such a star has, in general, a non-zero
**     proper motion in the FK5 system, the function requires the date
**     at which the position in the FK5 system was determined.
**
**  2) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  3) The FK5 to Hipparcos transformation is modeled as a pure
**     rotation and spin;  zonal errors in the FK5 catalog are not
**     taken into account.
**
**  4) The position returned by this function is in the Hipparcos
**     reference system but at date date1+date2.
**
**  5) See also iauFk52h, iauH2fk5, iauHfk5z.
**
**  Called:
**     iauS2c       spherical coordinates to unit vector
**     iauFk5hip    FK5 to Hipparcos rotation and spin
**     iauSxp       multiply p-vector by scalar
**     iauRv2m      r-vector to r-matrix
**     iauTrxp      product of transpose of r-matrix and p-vector
**     iauPxp       vector product of two p-vectors
**     iauC2s       p-vector to spherical
**     iauAnp       normalize angle into range 0 to 2pi
**
**  Reference:
```

```
**
**      F.Mignard & M.Froeschle, 2000, Astron.Astrophys. 354, 732-739.
**
*/
```

```
void iauFw2m(double gamb, double phib, double psi, double eps,
             double r[3][3])
/*
**  - - - - - - - -
**   i a u F w 2 m
**  - - - - - - - -
**
**  Form rotation matrix given the Fukushima-Williams angles.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     gamb     double         F-W angle gamma_bar (radians)
**     phib     double         F-W angle phi_bar (radians)
**     psi      double         F-W angle psi (radians)
**     eps      double         F-W angle epsilon (radians)
**
**  Returned:
**     r        double[3][3]   rotation matrix
**
**  Notes:
**
**  1) Naming the following points:
**
**           e = J2000.0 ecliptic pole,
**           p = GCRS pole,
**           E = ecliptic pole of date,
**     and   P = CIP,
**
**     the four Fukushima-Williams angles are as follows:
**
**        gamb = gamma = epE
**        phib = phi = pE
**        psi = psi = pEP
**        eps = epsilon = EP
**
**  2) The matrix representing the combined effects of frame bias,
**     precession and nutation is:
**
**        NxPxB = R_1(-eps).R_3(-psi).R_1(phib).R_3(gamb)
**
**  3) The present function can construct three different matrices,
**     depending on which angles are supplied as the arguments gamb,
**     phib, psi and eps:
**
**     o  To obtain the nutation x precession x frame bias matrix,
**        first generate the four precession angles known conventionally
**        as gamma_bar, phi_bar, psi_bar and epsilon_A, then generate
**        the nutation components Dpsi and Depsilon and add them to
**        psi_bar and epsilon_A, and finally call the present function
**        using those four angles as arguments.
**
**     o  To obtain the precession x frame bias matrix, generate the
**        four precession angles and call the present function.
**
**     o  To obtain the frame bias matrix, generate the four precession
**        angles for date J2000.0 and call the present function.
**
**     The nutation-only and precession-only matrices can if necessary
**     be obtained by combining these three appropriately.
**
**  Called:
**     iauIr        initialize r-matrix to identity
**     iauRz        rotate around Z-axis
**     iauRx        rotate around X-axis
**
**  References:
**
```

```
**      Capitaine, N. & Wallace, P.T., 2006, Astron.Astrophys. 450, 855
**
**      Hilton, J. et al., 2006, Celest.Mech.Dyn.Astron. 94, 351
**
*/
```

```
void iauFw2xy(double gamb, double phib, double psi, double eps,
              double *x, double *y)
/*
**  - - - - - - - - - -
**   i a u F w 2 x y
**  - - - - - - - - - -
**
**  CIP X,Y given Fukushima-Williams bias-precession-nutation angles.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     gamb     double    F-W angle gamma_bar (radians)
**     phib     double    F-W angle phi_bar (radians)
**     psi      double    F-W angle psi (radians)
**     eps      double    F-W angle epsilon (radians)
**
**  Returned:
**     x,y      double    CIP unit vector X,Y
**
**  Notes:
**
**  1) Naming the following points:
**
**           e = J2000.0 ecliptic pole,
**           p = GCRS pole
**           E = ecliptic pole of date,
**     and   P = CIP,
**
**     the four Fukushima-Williams angles are as follows:
**
**        gamb = gamma = epE
**        phib = phi = pE
**        psi = psi = pEP
**        eps = epsilon = EP
**
**  2) The matrix representing the combined effects of frame bias,
**     precession and nutation is:
**
**        NxPxB = R_1(-epsA).R_3(-psi).R_1(phib).R_3(gamb)
**
**     The returned values x,y are elements [2][0] and [2][1] of the
**     matrix.  Near J2000.0, they are essentially angles in radians.
**
**  Called:
**     iauFw2m      F-W angles to r-matrix
**     iauBpn2xy    extract CIP X,Y coordinates from NPB matrix
**
**  Reference:
**
**     Hilton, J. et al., 2006, Celest.Mech.Dyn.Astron. 94, 351
**
*/
```

```
void iauG2icrs ( double dl, double db, double *dr, double *dd )
/*
**  - - - - - - - - - -
**   i a u G 2 i c r s
**  - - - - - - - - - -
**
**  Transformation from Galactic coordinates to ICRS.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     dl     double      Galactic longitude (radians)
**     db     double      Galactic latitude (radians)
**
**  Returned:
**     dr     double      ICRS right ascension (radians)
**     dd     double      ICRS declination (radians)
**
**  Notes:
**
**  1) The IAU 1958 system of Galactic coordinates was defined with
**     respect to the now obsolete reference system FK4 B1950.0.  When
**     interpreting the system in a modern context, several factors have
**     to be taken into account:
**
**     . The inclusion in FK4 positions of the E-terms of aberration.
**
**     . The distortion of the FK4 proper motion system by differential
**       Galactic rotation.
**
**     . The use of the B1950.0 equinox rather than the now-standard
**       J2000.0.
**
**     . The frame bias between ICRS and the J2000.0 mean place system.
**
**     The Hipparcos Catalogue (Perryman & ESA 1997) provides a rotation
**     matrix that transforms directly between ICRS and Galactic
**     coordinates with the above factors taken into account.  The
**     matrix is derived from three angles, namely the ICRS coordinates
**     of the Galactic pole and the longitude of the ascending node of
**     the Galactic equator on the ICRS equator.  They are given in
**     degrees to five decimal places and for canonical purposes are
**     regarded as exact.  In the Hipparcos Catalogue the matrix
**     elements are given to 10 decimal places (about 20 microarcsec).
**     In the present SOFA function the matrix elements have been
**     recomputed from the canonical three angles and are given to 30
**     decimal places.
**
**  2) The inverse transformation is performed by the function iauIcrs2g.
**
**  Called:
**     iauAnp       normalize angle into range 0 to 2pi
**     iauAnpm      normalize angle into range +/- pi
**     iauS2c       spherical coordinates to unit vector
**     iauTrxp      product of transpose of r-matrix and p-vector
**     iauC2s       p-vector to spherical
**
**  Reference:
**     Perryman M.A.C. & ESA, 1997, ESA SP-1200, The Hipparcos and Tycho
**     catalogues.  Astrometric and photometric star catalogues
**     derived from the ESA Hipparcos Space Astrometry Mission.  ESA
**     Publications Division, Noordwijk, Netherlands.
**
*/
```

```
int iauGc2gd ( int n, double xyz[3],
               double *elong, double *phi, double *height )
/*
**  - - - - - - - - -
**   i a u G c 2 g d
**  - - - - - - - - -
**
**  Transform geocentric coordinates to geodetic using the specified
**  reference ellipsoid.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical transformation.
**
**  Given:
**     n       int        ellipsoid identifier (Note 1)
**     xyz     double[3]  geocentric vector (Note 2)
**
**  Returned:
**     elong   double     longitude (radians, east +ve, Note 3)
**     phi     double     latitude (geodetic, radians, Note 3)
**     height  double     height above ellipsoid (geodetic, Notes 2,3)
**
**  Returned (function value):
**            int         status:  0 = OK
**                                -1 = illegal identifier (Note 3)
**                                -2 = internal error (Note 3)
**
**  Notes:
**
**  1) The identifier n is a number that specifies the choice of
**     reference ellipsoid.  The following are supported:
**
**        n    ellipsoid
**
**        1     WGS84
**        2     GRS80
**        3     WGS72
**
**     The n value has no significance outside the SOFA software.  For
**     convenience, symbols WGS84 etc. are defined in sofam.h.
**
**  2) The geocentric vector (xyz, given) and height (height, returned)
**     are in meters.
**
**  3) An error status -1 means that the identifier n is illegal.  An
**     error status -2 is theoretically impossible.  In all error cases,
**     all three results are set to -1e9.
**
**  4) The inverse transformation is performed in the function iauGd2gc.
**
**  Called:
**     iauEform     Earth reference ellipsoids
**     iauGc2gde    geocentric to geodetic transformation, general
**
*/
```

```
int iauGc2gde ( double a, double f, double xyz[3],
                double *elong, double *phi, double *height )
/*
**  - - - - - - - - - -
**   i a u G c 2 g d e
**  - - - - - - - - - -
**
**  Transform geocentric coordinates to geodetic for a reference
**  ellipsoid of specified form.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     a       double     equatorial radius (Notes 2,4)
**     f       double     flattening (Note 3)
**     xyz     double[3]  geocentric vector (Note 4)
**
**  Returned:
**     elong   double     longitude (radians, east +ve)
**     phi     double     latitude (geodetic, radians)
**     height  double     height above ellipsoid (geodetic, Note 4)
**
**  Returned (function value):
**             int        status:  0 = OK
**                                -1 = illegal f
**                                -2 = illegal a
**
**  Notes:
**
**  1) This function is based on the GCONV2H Fortran subroutine by
**     Toshio Fukushima (see reference).
**
**  2) The equatorial radius, a, can be in any units, but meters is
**     the conventional choice.
**
**  3) The flattening, f, is (for the Earth) a value around 0.00335,
**     i.e. around 1/298.
**
**  4) The equatorial radius, a, and the geocentric vector, xyz,
**     must be given in the same units, and determine the units of
**     the returned height, height.
**
**  5) If an error occurs (status < 0), elong, phi and height are
**     unchanged.
**
**  6) The inverse transformation is performed in the function
**     iauGd2gce.
**
**  7) The transformation for a standard ellipsoid (such as WGS84) can
**     more conveniently be performed by calling iauGc2gd, which uses a
**     numerical code to identify the required A and F values.
**
**  Reference:
**
**     Fukushima, T., "Transformation from Cartesian to geodetic
**     coordinates accelerated by Halley's method", J.Geodesy (2006)
**     79: 689-693
**
*/
```

```
int iauGd2gc ( int n, double elong, double phi, double height,
               double xyz[3] )
/*
**  - - - - - - - - -
**   i a u G d 2 g c
**  - - - - - - - - -
**
**  Transform geodetic coordinates to geocentric using the specified
**  reference ellipsoid.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical transformation.
**
**  Given:
**     n       int        ellipsoid identifier (Note 1)
**     elong   double     longitude (radians, east +ve, Note 3)
**     phi     double     latitude (geodetic, radians, Note 3)
**     height  double     height above ellipsoid (geodetic, Notes 2,3)
**
**  Returned:
**     xyz     double[3]  geocentric vector (Note 2)
**
**  Returned (function value):
**             int        status:  0 = OK
**                                -1 = illegal identifier (Note 3)
**                                -2 = illegal case (Note 3)
**
**  Notes:
**
**  1) The identifier n is a number that specifies the choice of
**     reference ellipsoid.  The following are supported:
**
**        n    ellipsoid
**
**        1     WGS84
**        2     GRS80
**        3     WGS72
**
**     The n value has no significance outside the SOFA software.  For
**     convenience, symbols WGS84 etc. are defined in sofam.h.
**
**  2) The height (height, given) and the geocentric vector (xyz,
**     returned) are in meters.
**
**  3) No validation is performed on the arguments elong, phi and
**     height.  An error status -1 means that the identifier n is
**     illegal.  An error status -2 protects against cases that would
**     lead to arithmetic exceptions.  In all error cases, xyz is set
**     to zeros.
**
**  4) The inverse transformation is performed in the function iauGc2gd.
**
**  Called:
**     iauEform     Earth reference ellipsoids
**     iauGd2gce    geodetic to geocentric transformation, general
**     iauZp        zero p-vector
**
*/
```

```
int iauGd2gce ( double a, double f, double elong, double phi,
                double height, double xyz[3] )
/*
**  - - - - - - - - - - -
**   i a u G d 2 g c e
**  - - - - - - - - - - -
**
**  Transform geodetic coordinates to geocentric for a reference
**  ellipsoid of specified form.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     a       double     equatorial radius (Notes 1,3,4)
**     f       double     flattening (Notes 2,4)
**     elong   double     longitude (radians, east +ve, Note 4)
**     phi     double     latitude (geodetic, radians, Note 4)
**     height  double     height above ellipsoid (geodetic, Notes 3,4)
**
**  Returned:
**     xyz     double[3]  geocentric vector (Note 3)
**
**  Returned (function value):
**             int        status:  0 = OK
**                                -1 = illegal case (Note 4)
**  Notes:
**
**  1) The equatorial radius, a, can be in any units, but meters is
**     the conventional choice.
**
**  2) The flattening, f, is (for the Earth) a value around 0.00335,
**     i.e. around 1/298.
**
**  3) The equatorial radius, a, and the height, height, must be
**     given in the same units, and determine the units of the
**     returned geocentric vector, xyz.
**
**  4) No validation is performed on individual arguments.  The error
**     status -1 protects against (unrealistic) cases that would lead
**     to arithmetic exceptions.  If an error occurs, xyz is unchanged.
**
**  5) The inverse transformation is performed in the function
**     iauGc2gde.
**
**  6) The transformation for a standard ellipsoid (such as WGS84) can
**     more conveniently be performed by calling iauGd2gc,  which uses a
**     numerical code to identify the required a and f values.
**
**  References:
**
**     Green, R.M., Spherical Astronomy, Cambridge University Press,
**     (1985) Section 4.5, p96.
**
**     Explanatory Supplement to the Astronomical Almanac,
**     P. Kenneth Seidelmann (ed), University Science Books (1992),
**     Section 4.22, p202.
**
*/
```

```
double iauGmst00(double uta, double utb, double tta, double ttb)
/*
**  - - - - - - - - - -
**   i a u G m s t 0 0
**  - - - - - - - - - -
**
**  Greenwich mean sidereal time (model consistent with IAU 2000
**  resolutions).
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical model.
**
**  Given:
**     uta,utb    double    UT1 as a 2-part Julian Date (Notes 1,2)
**     tta,ttb    double    TT as a 2-part Julian Date (Notes 1,2)
**
**  Returned (function value):
**                double    Greenwich mean sidereal time (radians)
**
**  Notes:
**
**  1) The UT1 and TT dates uta+utb and tta+ttb respectively, are both
**     Julian Dates, apportioned in any convenient way between the
**     argument pairs.  For example, JD(UT1)=2450123.7 could be
**     expressed in any of these ways, among others:
**
**            Part A         Part B
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable (in the case of UT;  the TT is not at all critical
**     in this respect).  The J2000 and MJD methods are good compromises
**     between resolution and convenience.  For UT, the date & time
**     method is best matched to the algorithm that is used by the Earth
**     Rotation Angle function, called internally:  maximum precision is
**     delivered when the uta argument is for 0hrs UT1 on the day in
**     question and the utb argument lies in the range 0 to 1, or vice
**     versa.
**
**  2) Both UT1 and TT are required, UT1 to predict the Earth rotation
**     and TT to predict the effects of precession.  If UT1 is used for
**     both purposes, errors of order 100 microarcseconds result.
**
**  3) This GMST is compatible with the IAU 2000 resolutions and must be
**     used only in conjunction with other IAU 2000 compatible
**     components such as precession-nutation and equation of the
**     equinoxes.
**
**  4) The result is returned in the range 0 to 2pi.
**
**  5) The algorithm is from Capitaine et al. (2003) and IERS
**     Conventions 2003.
**
**  Called:
**     iauEra00     Earth rotation angle, IAU 2000
**     iauAnp       normalize angle into range 0 to 2pi
**
**  References:
**
**     Capitaine, N., Wallace, P.T. and McCarthy, D.D., "Expressions to
**     implement the IAU 2000 definition of UT1", Astronomy &
**     Astrophysics, 406, 1135-1149 (2003)
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
```

```
**      IERS Technical Note No. 32, BKG (2004)
**
*/
```

```
double iauGmst06(double uta, double utb, double tta, double ttb)
/*
**  - - - - - - - - - -
**   i a u G m s t 0 6
**  - - - - - - - - - -
**
**  Greenwich mean sidereal time (consistent with IAU 2006 precession).
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical model.
**
**  Given:
**     uta,utb    double    UT1 as a 2-part Julian Date (Notes 1,2)
**     tta,ttb    double    TT as a 2-part Julian Date (Notes 1,2)
**
**  Returned (function value):
**                double    Greenwich mean sidereal time (radians)
**
**  Notes:
**
**  1) The UT1 and TT dates uta+utb and tta+ttb respectively, are both
**     Julian Dates, apportioned in any convenient way between the
**     argument pairs.  For example, JD=2450123.7 could be expressed in
**     any of these ways, among others:
**
**            Part A        Part B
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable (in the case of UT;  the TT is not at all critical
**     in this respect).  The J2000 and MJD methods are good compromises
**     between resolution and convenience.  For UT, the date & time
**     method is best matched to the algorithm that is used by the Earth
**     rotation angle function, called internally:  maximum precision is
**     delivered when the uta argument is for 0hrs UT1 on the day in
**     question and the utb argument lies in the range 0 to 1, or vice
**     versa.
**
**  2) Both UT1 and TT are required, UT1 to predict the Earth rotation
**     and TT to predict the effects of precession.  If UT1 is used for
**     both purposes, errors of order 100 microarcseconds result.
**
**  3) This GMST is compatible with the IAU 2006 precession and must not
**     be used with other precession models.
**
**  4) The result is returned in the range 0 to 2pi.
**
**  Called:
**     iauEra00     Earth rotation angle, IAU 2000
**     iauAnp       normalize angle into range 0 to 2pi
**
**  Reference:
**
**     Capitaine, N., Wallace, P.T. & Chapront, J., 2005,
**     Astron.Astrophys. 432, 355
**
*/
```

```
double iauGmst82(double dj1, double dj2)
/*
**  - - - - - - - - - -
**   i a u G m s t 8 2
**  - - - - - - - - - -
**
**  Universal Time to Greenwich mean sidereal time (IAU 1982 model).
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical model.
**
**  Given:
**     dj1,dj2    double    UT1 Julian Date (see note)
**
**  Returned (function value):
**                double    Greenwich mean sidereal time (radians)
**
**  Notes:
**
**  1) The UT1 date dj1+dj2 is a Julian Date, apportioned in any
**     convenient way between the arguments dj1 and dj2.  For example,
**     JD(UT1)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**             dj1            dj2
**
**         2450123.7          0          (JD method)
**          2451545        -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5         0.2         (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 and MJD methods are good compromises
**     between resolution and convenience.  The date & time method is
**     best matched to the algorithm used:  maximum accuracy (or, at
**     least, minimum noise) is delivered when the dj1 argument is for
**     0hrs UT1 on the day in question and the dj2 argument lies in the
**     range 0 to 1, or vice versa.
**
**  2) The algorithm is based on the IAU 1982 expression.  This is
**     always described as giving the GMST at 0 hours UT1.  In fact, it
**     gives the difference between the GMST and the UT, the steady
**     4-minutes-per-day drawing-ahead of ST with respect to UT.  When
**     whole days are ignored, the expression happens to equal the GMST
**     at 0 hours UT1 each day.
**
**  3) In this function, the entire UT1 (the sum of the two arguments
**     dj1 and dj2) is used directly as the argument for the standard
**     formula, the constant term of which is adjusted by 12 hours to
**     take account of the noon phasing of Julian Date.  The UT1 is then
**     added, but omitting whole days to conserve accuracy.
**
**  Called:
**     iauAnp       normalize angle into range 0 to 2pi
**
**  References:
**
**     Transactions of the International Astronomical Union,
**     XVIII B, 67 (1983).
**
**     Aoki et al., Astron.Astrophys., 105, 359-361 (1982).
**
*/
```

```
double iauGst00a(double uta, double utb, double tta, double ttb)
/*
**  - - - - - - - - - -
**   i a u G s t 0 0 a
**  - - - - - - - - - -
**
**  Greenwich apparent sidereal time (consistent with IAU 2000
**  resolutions).
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical model.
**
**  Given:
**     uta,utb    double    UT1 as a 2-part Julian Date (Notes 1,2)
**     tta,ttb    double    TT as a 2-part Julian Date (Notes 1,2)
**
**  Returned (function value):
**                double    Greenwich apparent sidereal time (radians)
**
**  Notes:
**
**  1) The UT1 and TT dates uta+utb and tta+ttb respectively, are both
**     Julian Dates, apportioned in any convenient way between the
**     argument pairs.  For example, JD(UT1)=2450123.7 could be
**     expressed in any of these ways, among others:
**
**             uta            utb
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable (in the case of UT;  the TT is not at all critical
**     in this respect).  The J2000 and MJD methods are good compromises
**     between resolution and convenience.  For UT, the date & time
**     method is best matched to the algorithm that is used by the Earth
**     Rotation Angle function, called internally:  maximum precision is
**     delivered when the uta argument is for 0hrs UT1 on the day in
**     question and the utb argument lies in the range 0 to 1, or vice
**     versa.
**
**  2) Both UT1 and TT are required, UT1 to predict the Earth rotation
**     and TT to predict the effects of precession-nutation.  If UT1 is
**     used for both purposes, errors of order 100 microarcseconds
**     result.
**
**  3) This GAST is compatible with the IAU 2000 resolutions and must be
**     used only in conjunction with other IAU 2000 compatible
**     components such as precession-nutation.
**
**  4) The result is returned in the range 0 to 2pi.
**
**  5) The algorithm is from Capitaine et al. (2003) and IERS
**     Conventions 2003.
**
**  Called:
**     iauGmst00    Greenwich mean sidereal time, IAU 2000
**     iauEe00a     equation of the equinoxes, IAU 2000A
**     iauAnp       normalize angle into range 0 to 2pi
**
**  References:
**
**     Capitaine, N., Wallace, P.T. and McCarthy, D.D., "Expressions to
**     implement the IAU 2000 definition of UT1", Astronomy &
**     Astrophysics, 406, 1135-1149 (2003)
**
```

```
**      McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**      IERS Technical Note No. 32, BKG (2004)
**
*/
```

```
double iauGst00b(double uta, double utb)
/*
**  - - - - - - - - - -
**   i a u G s t 0 0 b
**  - - - - - - - - - -
**
**  Greenwich apparent sidereal time (consistent with IAU 2000
**  resolutions but using the truncated nutation model IAU 2000B).
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     uta,utb    double    UT1 as a 2-part Julian Date (Notes 1,2)
**
**  Returned (function value):
**                double    Greenwich apparent sidereal time (radians)
**
**  Notes:
**
**  1) The UT1 date uta+utb is a Julian Date, apportioned in any
**     convenient way between the argument pair.  For example,
**     JD(UT1)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**             uta            utb
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in cases
**     where the loss of several decimal digits of resolution is
**     acceptable.  The J2000 and MJD methods are good compromises
**     between resolution and convenience.  For UT, the date & time
**     method is best matched to the algorithm that is used by the Earth
**     Rotation Angle function, called internally:  maximum precision is
**     delivered when the uta argument is for 0hrs UT1 on the day in
**     question and the utb argument lies in the range 0 to 1, or vice
**     versa.
**
**  2) The result is compatible with the IAU 2000 resolutions, except
**     that accuracy has been compromised for the sake of speed and
**     convenience in two respects:
**
**     . UT is used instead of TDB (or TT) to compute the precession
**       component of GMST and the equation of the equinoxes.  This
**       results in errors of order 0.1 mas at present.
**
**     . The IAU 2000B abridged nutation model (McCarthy & Luzum, 2003)
**       is used, introducing errors of up to 1 mas.
**
**  3) This GAST is compatible with the IAU 2000 resolutions and must be
**     used only in conjunction with other IAU 2000 compatible
**     components such as precession-nutation.
**
**  4) The result is returned in the range 0 to 2pi.
**
**  5) The algorithm is from Capitaine et al. (2003) and IERS
**     Conventions 2003.
**
**  Called:
**     iauGmst00    Greenwich mean sidereal time, IAU 2000
**     iauEe00b     equation of the equinoxes, IAU 2000B
**     iauAnp       normalize angle into range 0 to 2pi
**
**  References:
**
```

```
**      Capitaine, N., Wallace, P.T. and McCarthy, D.D., "Expressions to
**      implement the IAU 2000 definition of UT1", Astronomy &
**      Astrophysics, 406, 1135-1149 (2003)
**
**      McCarthy, D.D. & Luzum, B.J., "An abridged model of the
**      precession-nutation of the celestial pole", Celestial Mechanics &
**      Dynamical Astronomy, 85, 37-49 (2003)
**
**      McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**      IERS Technical Note No. 32, BKG (2004)
**
*/
```

```
double iauGst06(double uta, double utb, double tta, double ttb,
                double rnpb[3][3])
/*
**  - - - - - - - - - -
**   i a u G s t 0 6
**  - - - - - - - - - -
**
**  Greenwich apparent sidereal time, IAU 2006, given the NPB matrix.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     uta,utb  double        UT1 as a 2-part Julian Date (Notes 1,2)
**     tta,ttb  double        TT as a 2-part Julian Date (Notes 1,2)
**     rnpb     double[3][3]  nutation x precession x bias matrix
**
**  Returned (function value):
**              double        Greenwich apparent sidereal time (radians)
**
**  Notes:
**
**  1) The UT1 and TT dates uta+utb and tta+ttb respectively, are both
**     Julian Dates, apportioned in any convenient way between the
**     argument pairs.  For example, JD(UT1)=2450123.7 could be
**     expressed in any of these ways, among others:
**
**             uta            utb
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable (in the case of UT;  the TT is not at all critical
**     in this respect).  The J2000 and MJD methods are good compromises
**     between resolution and convenience.  For UT, the date & time
**     method is best matched to the algorithm that is used by the Earth
**     rotation angle function, called internally:  maximum precision is
**     delivered when the uta argument is for 0hrs UT1 on the day in
**     question and the utb argument lies in the range 0 to 1, or vice
**     versa.
**
**  2) Both UT1 and TT are required, UT1 to predict the Earth rotation
**     and TT to predict the effects of precession-nutation.  If UT1 is
**     used for both purposes, errors of order 100 microarcseconds
**     result.
**
**  3) Although the function uses the IAU 2006 series for s+XY/2, it is
**     otherwise independent of the precession-nutation model and can in
**     practice be used with any equinox-based NPB matrix.
**
**  4) The result is returned in the range 0 to 2pi.
**
**  Called:
**     iauBpn2xy    extract CIP X,Y coordinates from NPB matrix
**     iauS06       the CIO locator s, given X,Y, IAU 2006
**     iauAnp       normalize angle into range 0 to 2pi
**     iauEra00     Earth rotation angle, IAU 2000
**     iauEors      equation of the origins, given NPB matrix and s
**
**  Reference:
**
**     Wallace, P.T. & Capitaine, N., 2006, Astron.Astrophys. 459, 981
**
*/
```

```
double iauGst06a(double uta, double utb, double tta, double ttb)
/*
**  - - - - - - - - - -
**   i a u G s t 0 6 a
**  - - - - - - - - - -
**
**  Greenwich apparent sidereal time (consistent with IAU 2000 and 2006
**  resolutions).
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical model.
**
**  Given:
**     uta,utb    double    UT1 as a 2-part Julian Date (Notes 1,2)
**     tta,ttb    double    TT as a 2-part Julian Date (Notes 1,2)
**
**  Returned (function value):
**                double    Greenwich apparent sidereal time (radians)
**
**  Notes:
**
**  1) The UT1 and TT dates uta+utb and tta+ttb respectively, are both
**     Julian Dates, apportioned in any convenient way between the
**     argument pairs.  For example, JD(UT1)=2450123.7 could be
**     expressed in any of these ways, among others:
**
**             uta            utb
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable (in the case of UT;  the TT is not at all critical
**     in this respect).  The J2000 and MJD methods are good compromises
**     between resolution and convenience.  For UT, the date & time
**     method is best matched to the algorithm that is used by the Earth
**     rotation angle function, called internally:  maximum precision is
**     delivered when the uta argument is for 0hrs UT1 on the day in
**     question and the utb argument lies in the range 0 to 1, or vice
**     versa.
**
**  2) Both UT1 and TT are required, UT1 to predict the Earth rotation
**     and TT to predict the effects of precession-nutation.  If UT1 is
**     used for both purposes, errors of order 100 microarcseconds
**     result.
**
**  3) This GAST is compatible with the IAU 2000/2006 resolutions and
**     must be used only in conjunction with IAU 2006 precession and
**     IAU 2000A nutation.
**
**  4) The result is returned in the range 0 to 2pi.
**
**  Called:
**     iauPnm06a    classical NPB matrix, IAU 2006/2000A
**     iauGst06     Greenwich apparent ST, IAU 2006, given NPB matrix
**
**  Reference:
**
**     Wallace, P.T. & Capitaine, N., 2006, Astron.Astrophys. 459, 981
**
*/
```

```
double iauGst94(double uta, double utb)
/*
**  - - - - - - - - -
**   i a u G s t 9 4
**  - - - - - - - - -
**
**  Greenwich apparent sidereal time (consistent with IAU 1982/94
**  resolutions).
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     uta,utb    double    UT1 as a 2-part Julian Date (Notes 1,2)
**
**  Returned (function value):
**                double    Greenwich apparent sidereal time (radians)
**
**  Notes:
**
**  1) The UT1 date uta+utb is a Julian Date, apportioned in any
**     convenient way between the argument pair.  For example,
**     JD(UT1)=2450123.7 could be expressed in any of these ways, among
**     others:
**
**             uta            utb
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in cases
**     where the loss of several decimal digits of resolution is
**     acceptable.  The J2000 and MJD methods are good compromises
**     between resolution and convenience.  For UT, the date & time
**     method is best matched to the algorithm that is used by the Earth
**     Rotation Angle function, called internally:  maximum precision is
**     delivered when the uta argument is for 0hrs UT1 on the day in
**     question and the utb argument lies in the range 0 to 1, or vice
**     versa.
**
**  2) The result is compatible with the IAU 1982 and 1994 resolutions,
**     except that accuracy has been compromised for the sake of
**     convenience in that UT is used instead of TDB (or TT) to compute
**     the equation of the equinoxes.
**
**  3) This GAST must be used only in conjunction with contemporaneous
**     IAU standards such as 1976 precession, 1980 obliquity and 1982
**     nutation.  It is not compatible with the IAU 2000 resolutions.
**
**  4) The result is returned in the range 0 to 2pi.
**
**  Called:
**     iauGmst82    Greenwich mean sidereal time, IAU 1982
**     iauEqeq94    equation of the equinoxes, IAU 1994
**     iauAnp       normalize angle into range 0 to 2pi
**
**  References:
**
**     Explanatory Supplement to the Astronomical Almanac,
**     P. Kenneth Seidelmann (ed), University Science Books (1992)
**
**     IAU Resolution C7, Recommendation 3 (1994)
**
*/
```

```
void iauH2fk5(double rh, double dh,
              double drh, double ddh, double pxh, double rvh,
              double *r5, double *d5,
              double *dr5, double *dd5, double *px5, double *rv5)
/*
**  - - - - - - - - -
**   i a u H 2 f k 5
**  - - - - - - - - -
**
**  Transform Hipparcos star data into the FK5 (J2000.0) system.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given (all Hipparcos, epoch J2000.0):
**     rh      double    RA (radians)
**     dh      double    Dec (radians)
**     drh     double    proper motion in RA (dRA/dt, rad/Jyear)
**     ddh     double    proper motion in Dec (dDec/dt, rad/Jyear)
**     pxh     double    parallax (arcsec)
**     rvh     double    radial velocity (km/s, positive = receding)
**
**  Returned (all FK5, equinox J2000.0, epoch J2000.0):
**     r5      double    RA (radians)
**     d5      double    Dec (radians)
**     dr5     double    proper motion in RA (dRA/dt, rad/Jyear)
**     dd5     double    proper motion in Dec (dDec/dt, rad/Jyear)
**     px5     double    parallax (arcsec)
**     rv5     double    radial velocity (km/s, positive = receding)
**
**  Notes:
**
**  1) This function transforms Hipparcos star positions and proper
**     motions into FK5 J2000.0.
**
**  2) The proper motions in RA are dRA/dt rather than
**     cos(Dec)*dRA/dt, and are per year rather than per century.
**
**  3) The FK5 to Hipparcos transformation is modeled as a pure
**     rotation and spin;  zonal errors in the FK5 catalog are not
**     taken into account.
**
**  4) See also iauFk52h, iauFk5hz, iauHfk5z.
**
**  Called:
**     iauStarpv    star catalog data to space motion pv-vector
**     iauFk5hip    FK5 to Hipparcos rotation and spin
**     iauRv2m      r-vector to r-matrix
**     iauRxp       product of r-matrix and p-vector
**     iauTrxp      product of transpose of r-matrix and p-vector
**     iauPxp       vector product of two p-vectors
**     iauPmp       p-vector minus p-vector
**     iauPvstar    space motion pv-vector to star catalog data
**
**  Reference:
**
**     F.Mignard & M.Froeschle, Astron.Astrophys., 354, 732-739 (2000).
**
*/
```

```
void iauHd2ae (double ha, double dec, double phi,
               double *az, double *el)
/*
**  - - - - - - - - -
**   i a u H d 2 a e
**  - - - - - - - - -
**
**  Equatorial to horizon coordinates:  transform hour angle and
**  declination to azimuth and altitude.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     ha       double       hour angle (local)
**     dec      double       declination
**     phi      double       site latitude
**
**  Returned:
**     *az      double       azimuth
**     *el      double       altitude (informally, elevation)
**
**  Notes:
**
**  1)  All the arguments are angles in radians.
**
**  2)  Azimuth is returned in the range 0-2pi;  north is zero, and east
**      is +pi/2.  Altitude is returned in the range +/- pi/2.
**
**  3)  The latitude phi is pi/2 minus the angle between the Earth's
**      rotation axis and the adopted zenith.  In many applications it
**      will be sufficient to use the published geodetic latitude of the
**      site.  In very precise (sub-arcsecond) applications, phi can be
**      corrected for polar motion.
**
**  4)  The returned azimuth az is with respect to the rotational north
**      pole, as opposed to the ITRS pole, and for sub-arcsecond
**      accuracy will need to be adjusted for polar motion if it is to
**      be with respect to north on a map of the Earth's surface.
**
**  5)  Should the user wish to work with respect to the astronomical
**      zenith rather than the geodetic zenith, phi will need to be
**      adjusted for deflection of the vertical (often tens of
**      arcseconds), and the zero point of the hour angle ha will also
**      be affected.
**
**  6)  The transformation is the same as Vh = Rz(pi)*Ry(pi/2-phi)*Ve,
**      where Vh and Ve are lefthanded unit vectors in the (az,el) and
**      (ha,dec) systems respectively and Ry and Rz are rotations about
**      first the y-axis and then the z-axis.  (n.b. Rz(pi) simply
**      reverses the signs of the x and y components.)  For efficiency,
**      the algorithm is written out rather than calling other utility
**      functions.  For applications that require even greater
**      efficiency, additional savings are possible if constant terms
**      such as functions of latitude are computed once and for all.
**
**  7)  Again for efficiency, no range checking of arguments is carried
**      out.
**
**  Last revision:   2021 February 24
**
**  SOFA release 2023-10-11
**
**  Copyright (C) 2023 IAU SOFA Board.  See notes at end.
*/
{
   double sh, ch, sd, cd, sp, cp, x, y, z, r, a;
```

```
/* Useful trig functions. */
   sh = sin(ha);
   ch = cos(ha);
   sd = sin(dec);
   cd = cos(dec);
   sp = sin(phi);
   cp = cos(phi);

/* Az,Alt unit vector. */
   x = - ch*cd*sp + sd*cp;
   y = - sh*cd;
   z = ch*cd*cp + sd*sp;

/* To spherical. */
   r = sqrt(x*x + y*y);
   a = (r != 0.0) ? atan2(y,x) : 0.0;
   *az = (a < 0.0) ? a+D2PI : a;
   *el = atan2(z,r);

/* Finished. */
```

```
**
**     Note that, as originally distributed, the SOFA software is
**     intended to be a definitive implementation of the IAU standards,
**     and consequently third-party modifications are discouraged.  All
**     variations, no matter how minor, must be explicitly marked as
**     such, as explained above.
**
**  4. You shall not cause the SOFA software to be brought into
**     disrepute, either by misuse, or use for inappropriate tasks, or
**     by inappropriate modification.
**
**  5. The SOFA software is provided "as is" and SOFA makes no warranty
**     as to its use or performance.   SOFA does not and cannot warrant
**     the performance or results which the user may obtain by using the
**     SOFA software.  SOFA makes no warranties, express or implied, as
**     to non-infringement of third party rights, merchantability, or
**     fitness for any particular purpose.  In no event will SOFA be
**     liable to the user for any consequential, incidental, or special
**     damages, including any lost profits or lost savings, even if a
**     SOFA representative has been advised of such damages, or for any
**     claim by any third party.
**
**  6. The provision of any version of the SOFA software under the terms
**     and conditions specified herein does not imply that future
**     versions will also be made available under the same terms and
**     conditions.
*
**  In any published work or commercial product which uses the SOFA
**  software directly, acknowledgement (see www.iausofa.org) is
**  appreciated.
**
**  Correspondence concerning SOFA software should be addressed as
**  follows:
**
**      By email:  sofa@ukho.gov.uk
**      By post:   IAU SOFA Center
**                 HM Nautical Almanac Office
**                 UK Hydrographic Office
**                 Admiralty Way, Taunton
**                 Somerset, TA1 2DN
**                 United Kingdom
**
**--------------------------------------------------------------------*/
}
```

```
double iauHd2pa (double ha, double dec, double phi)
/*
**  - - - - - - - - -
**   i a u H d 2 p a
**  - - - - - - - - -
**
**  Parallactic angle for a given hour angle and declination.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     ha     double     hour angle
**     dec    double     declination
**     phi    double     site latitude
**
**  Returned (function value):
**            double     parallactic angle
**
**  Notes:
**
**  1)  All the arguments are angles in radians.
**
**  2)  The parallactic angle at a point in the sky is the position
**      angle of the vertical, i.e. the angle between the directions to
**      the north celestial pole and to the zenith respectively.
**
**  3)  The result is returned in the range -pi to +pi.
**
**  4)  At the pole itself a zero result is returned.
**
**  5)  The latitude phi is pi/2 minus the angle between the Earth's
**      rotation axis and the adopted zenith.  In many applications it
**      will be sufficient to use the published geodetic latitude of the
**      site.  In very precise (sub-arcsecond) applications, phi can be
**      corrected for polar motion.
**
**  6)  Should the user wish to work with respect to the astronomical
**      zenith rather than the geodetic zenith, phi will need to be
**      adjusted for deflection of the vertical (often tens of
**      arcseconds), and the zero point of the hour angle ha will also
**      be affected.
**
**  Reference:
**     Smart, W.M., "Spherical Astronomy", Cambridge University Press,
**     6th edition (Green, 1977), p49.
**
**  Last revision:   2017 September 12
**
**  SOFA release 2023-10-11
**
**  Copyright (C) 2023 IAU SOFA Board.  See notes at end.
*/
{
   double cp, cqsz, sqsz;


   cp = cos(phi);
   sqsz = cp*sin(ha);
   cqsz = sin(phi)*cos(dec) - cp*sin(dec)*cos(ha);
   return ( ( sqsz != 0.0 || cqsz != 0.0 ) ? atan2(sqsz,cqsz) : 0.0 );

/* Finished. */

/*----------------------------------------------------------------------
**
**  Copyright (C) 2023
**  Standards of Fundamental Astronomy Board
**  of the International Astronomical Union.
```

```
*
**  In any published work or commercial product which uses the SOFA
**  software directly, acknowledgement (see www.iausofa.org) is
**  appreciated.
**
**  Correspondence concerning SOFA software should be addressed as
**  follows:
**
**      By email:  sofa@ukho.gov.uk
**      By post:   IAU SOFA Center
**                 HM Nautical Almanac Office
**                 UK Hydrographic Office
**                 Admiralty Way, Taunton
**                 Somerset, TA1 2DN
**                 United Kingdom
**
**--------------------------------------------------------------------*/
}
```

```
void iauHfk5z(double rh, double dh, double date1, double date2,
              double *r5, double *d5, double *dr5, double *dd5)
/*
**  - - - - - - - - -
**   i a u H f k 5 z
**  - - - - - - - - -
**
**  Transform a Hipparcos star position into FK5 J2000.0, assuming
**  zero Hipparcos proper motion.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     rh            double    Hipparcos RA (radians)
**     dh            double    Hipparcos Dec (radians)
**     date1,date2   double    TDB date (Note 1)
**
**  Returned (all FK5, equinox J2000.0, date date1+date2):
**     r5            double    RA (radians)
**     d5            double    Dec (radians)
**     dr5           double    RA proper motion (rad/year, Note 4)
**     dd5           double    Dec proper motion (rad/year, Note 4)
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The proper motion in RA is dRA/dt rather than cos(Dec)*dRA/dt.
**
**  3) The FK5 to Hipparcos transformation is modeled as a pure rotation
**     and spin;  zonal errors in the FK5 catalog are not taken into
**     account.
**
**  4) It was the intention that Hipparcos should be a close
**     approximation to an inertial frame, so that distant objects have
**     zero proper motion;  such objects have (in general) non-zero
**     proper motion in FK5, and this function returns those fictitious
**     proper motions.
**
**  5) The position returned by this function is in the FK5 J2000.0
**     reference system but at date date1+date2.
**
**  6) See also iauFk52h, iauH2fk5, iauFk5hz.
**
**  Called:
**     iauS2c       spherical coordinates to unit vector
**     iauFk5hip    FK5 to Hipparcos rotation and spin
**     iauRxp       product of r-matrix and p-vector
**     iauSxp       multiply p-vector by scalar
**     iauRxr       product of two r-matrices
**     iauTrxp      product of transpose of r-matrix and p-vector
```

```
**     iauPxp       vector product of two p-vectors
**     iauPv2s      pv-vector to spherical
**     iauAnp       normalize angle into range 0 to 2pi
**
**  Reference:
**
**     F.Mignard & M.Froeschle, 2000, Astron.Astrophys. 354, 732-739.
**
*/
```

```
void iauIcrs2g ( double dr, double dd, double *dl, double *db )
/*
**  - - - - - - - - - -
**   i a u I c r s 2 g
**  - - - - - - - - - -
**
**  Transformation from ICRS to Galactic coordinates.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     dr     double      ICRS right ascension (radians)
**     dd     double      ICRS declination (radians)
**
**  Returned:
**     dl     double      Galactic longitude (radians)
**     db     double      Galactic latitude (radians)
**
**  Notes:
**
**  1) The IAU 1958 system of Galactic coordinates was defined with
**     respect to the now obsolete reference system FK4 B1950.0.  When
**     interpreting the system in a modern context, several factors have
**     to be taken into account:
**
**     . The inclusion in FK4 positions of the E-terms of aberration.
**
**     . The distortion of the FK4 proper motion system by differential
**       Galactic rotation.
**
**     . The use of the B1950.0 equinox rather than the now-standard
**       J2000.0.
**
**     . The frame bias between ICRS and the J2000.0 mean place system.
**
**     The Hipparcos Catalogue (Perryman & ESA 1997) provides a rotation
**     matrix that transforms directly between ICRS and Galactic
**     coordinates with the above factors taken into account.  The
**     matrix is derived from three angles, namely the ICRS coordinates
**     of the Galactic pole and the longitude of the ascending node of
**     the Galactic equator on the ICRS equator.  They are given in
**     degrees to five decimal places and for canonical purposes are
**     regarded as exact.  In the Hipparcos Catalogue the matrix
**     elements are given to 10 decimal places (about 20 microarcsec).
**     In the present SOFA function the matrix elements have been
**     recomputed from the canonical three angles and are given to 30
**     decimal places.
**
**  2) The inverse transformation is performed by the function iauG2icrs.
**
**  Called:
**     iauAnp       normalize angle into range 0 to 2pi
**     iauAnpm      normalize angle into range +/- pi
**     iauS2c       spherical coordinates to unit vector
**     iauRxp       product of r-matrix and p-vector
**     iauC2s       p-vector to spherical
**
**  Reference:
**     Perryman M.A.C. & ESA, 1997, ESA SP-1200, The Hipparcos and Tycho
**     catalogues.  Astrometric and photometric star catalogues
**     derived from the ESA Hipparcos Space Astrometry Mission.  ESA
**     Publications Division, Noordwijk, Netherlands.
**
*/
```

```
void iauIr(double r[3][3])
/*
**  - - - - - -
**   i a u I r
**  - - - - - -
**
**  Initialize an r-matrix to the identity matrix.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Returned:
**     r       double[3][3]    r-matrix
**
*/
```

```
int iauJd2cal(double dj1, double dj2,
              int *iy, int *im, int *id, double *fd)
/*
**  - - - - - - - - - -
**   i a u J d 2 c a l
**  - - - - - - - - - -
**
**  Julian Date to Gregorian year, month, day, and fraction of a day.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     dj1,dj2   double   Julian Date (Notes 1, 2)
**
**  Returned (arguments):
**     iy        int      year
**     im        int      month
**     id        int      day
**     fd        double   fraction of day
**
**  Returned (function value):
**               int      status:
**                           0 = OK
**                          -1 = unacceptable date (Note 1)
**
**  Notes:
**
**  1) The earliest valid date is -68569.5 (-4900 March 1).  The
**     largest value accepted is 1e9.
**
**  2) The Julian Date is apportioned in any convenient way between
**     the arguments dj1 and dj2.  For example, JD=2450123.7 could
**     be expressed in any of these ways, among others:
**
**            dj1             dj2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     Separating integer and fraction uses the "compensated summation"
**     algorithm of Kahan-Neumaier to preserve as much precision as
**     possible irrespective of the jd1+jd2 apportionment.
**
**  3) In early eras the conversion is from the "proleptic Gregorian
**     calendar";  no account is taken of the date(s) of adoption of
**     the Gregorian calendar, nor is the AD/BC numbering convention
**     observed.
**
**  References:
**
**     Explanatory Supplement to the Astronomical Almanac,
**     P. Kenneth Seidelmann (ed), University Science Books (1992),
**     Section 12.92 (p604).
**
**     Klein, A., A Generalized Kahan-Babuska-Summation-Algorithm.
**     Computing, 76, 279-293 (2006), Section 3.
**
*/
```

```
int iauJdcalf(int ndp, double dj1, double dj2, int iymdf[4])
/*
**  - - - - - - - - - -
**   i a u J d c a l f
**  - - - - - - - - - -
**
**  Julian Date to Gregorian Calendar, expressed in a form convenient
**  for formatting messages:  rounded to a specified precision.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     ndp       int      number of decimal places of days in fraction
**     dj1,dj2   double   dj1+dj2 = Julian Date (Note 1)
**
**  Returned:
**     iymdf     int[4]   year, month, day, fraction in Gregorian
**                        calendar
**
**  Returned (function value):
**               int      status:
**                          -1 = date out of range
**                           0 = OK
**                          +1 = ndp not 0-9 (interpreted as 0)
**
**  Notes:
**
**  1) The Julian Date is apportioned in any convenient way between
**     the arguments dj1 and dj2.  For example, JD=2450123.7 could
**     be expressed in any of these ways, among others:
**
**             dj1            dj2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**  2) In early eras the conversion is from the "Proleptic Gregorian
**     Calendar";  no account is taken of the date(s) of adoption of
**     the Gregorian Calendar, nor is the AD/BC numbering convention
**     observed.
**
**  3) See also the function iauJd2cal.
**
**  4) The number of decimal places ndp should be 4 or less if internal
**     overflows are to be avoided on platforms which use 16-bit
**     integers.
**
**  Called:
**     iauJd2cal    JD to Gregorian calendar
**
**  Reference:
**
**     Explanatory Supplement to the Astronomical Almanac,
**     P. Kenneth Seidelmann (ed), University Science Books (1992),
**     Section 12.92 (p604).
**
*/
```

```
void iauLd(double bm, double p[3], double q[3], double e[3],
           double em, double dlim, double p1[3])
/*
**  - - - - - -
**   i a u L d
**  - - - - - -
**
**  Apply light deflection by a solar-system body, as part of
**  transforming coordinate direction into natural direction.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     bm     double     mass of the gravitating body (solar masses)
**     p      double[3]  direction from observer to source (unit vector)
**     q      double[3]  direction from body to source (unit vector)
**     e      double[3]  direction from body to observer (unit vector)
**     em     double     distance from body to observer (au)
**     dlim   double     deflection limiter (Note 4)
**
**  Returned:
**     p1     double[3]  observer to deflected source (unit vector)
**
**  Notes:
**
**  1) The algorithm is based on Expr. (70) in Klioner (2003) and
**     Expr. (7.63) in the Explanatory Supplement (Urban & Seidelmann
**     2013), with some rearrangement to minimize the effects of machine
**     precision.
**
**  2) The mass parameter bm can, as required, be adjusted in order to
**     allow for such effects as quadrupole field.
**
**  3) The barycentric position of the deflecting body should ideally
**     correspond to the time of closest approach of the light ray to
**     the body.
**
**  4) The deflection limiter parameter dlim is phi^2/2, where phi is
**     the angular separation (in radians) between source and body at
**     which limiting is applied.  As phi shrinks below the chosen
**     threshold, the deflection is artificially reduced, reaching zero
**     for phi = 0.
**
**  5) The returned vector p1 is not normalized, but the consequential
**     departure from unit magnitude is always negligible.
**
**  6) The arguments p and p1 can be the same array.
**
**  7) To accumulate total light deflection taking into account the
**     contributions from several bodies, call the present function for
**     each body in succession, in decreasing order of distance from the
**     observer.
**
**  8) For efficiency, validation is omitted.  The supplied vectors must
**     be of unit magnitude, and the deflection limiter non-zero and
**     positive.
**
**  References:
**
**     Urban, S. & Seidelmann, P. K. (eds), Explanatory Supplement to
**     the Astronomical Almanac, 3rd ed., University Science Books
**     (2013).
**
**     Klioner, Sergei A., "A practical relativistic model for micro-
**     arcsecond astrometry in space", Astr. J. 125, 1580-1597 (2003).
**
**  Called:
**     iauPdp       scalar product of two p-vectors
```

```
**     iauPxp       vector product of two p-vectors
**
*/
```

```
void iauLdn(int n, iauLDBODY b[], double ob[3], double sc[3],
            double sn[3])
/*+
**  - - - - - - -
**   i a u L d n
**  - - - - - - -
**
**  For a star, apply light deflection by multiple solar-system bodies,
**  as part of transforming coordinate direction into natural direction.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     n    int           number of bodies (note 1)
**     b    iauLDBODY[n]  data for each of the n bodies (Notes 1,2):
**      bm   double         mass of the body (solar masses, Note 3)
**      dl   double         deflection limiter (Note 4)
**      pv   [2][3]         barycentric PV of the body (au, au/day)
**     ob   double[3]     barycentric position of the observer (au)
**     sc   double[3]     observer to star coord direction (unit vector)
**
**  Returned:
**     sn    double[3]      observer to deflected star (unit vector)
**
**  1) The array b contains n entries, one for each body to be
**     considered.  If n = 0, no gravitational light deflection will be
**     applied, not even for the Sun.
**
**  2) The array b should include an entry for the Sun as well as for
**     any planet or other body to be taken into account.  The entries
**     should be in the order in which the light passes the body.
**
**  3) In the entry in the b array for body i, the mass parameter
**     b[i].bm can, as required, be adjusted in order to allow for such
**     effects as quadrupole field.
**
**  4) The deflection limiter parameter b[i].dl is phi^2/2, where phi is
**     the angular separation (in radians) between star and body at
**     which limiting is applied.  As phi shrinks below the chosen
**     threshold, the deflection is artificially reduced, reaching zero
**     for phi = 0.   Example values suitable for a terrestrial
**     observer, together with masses, are as follows:
**
**        body i     b[i].bm        b[i].dl
**
**        Sun        1.0            6e-6
**        Jupiter    0.00095435     3e-9
**        Saturn     0.00028574     3e-10
**
**  5) For cases where the starlight passes the body before reaching the
**     observer, the body is placed back along its barycentric track by
**     the light time from that point to the observer.  For cases where
**     the body is "behind" the observer no such shift is applied.  If
**     a different treatment is preferred, the user has the option of
**     instead using the iauLd function.  Similarly, iauLd can be used
**     for cases where the source is nearby, not a star.
**
**  6) The returned vector sn is not normalized, but the consequential
**     departure from unit magnitude is always negligible.
**
**  7) The arguments sc and sn can be the same array.
**
**  8) For efficiency, validation is omitted.  The supplied masses must
**     be greater than zero, the position and velocity vectors must be
**     right, and the deflection limiter greater than zero.
**
**  Reference:
**
```

```
**     Urban, S. & Seidelmann, P. K. (eds), Explanatory Supplement to
**     the Astronomical Almanac, 3rd ed., University Science Books
**     (2013), Section 7.2.4.
**
**  Called:
**     iauCp        copy p-vector
**     iauPdp       scalar product of two p-vectors
**     iauPmp       p-vector minus p-vector
**     iauPpsp      p-vector plus scaled p-vector
**     iauPn        decompose p-vector into modulus and direction
**     iauLd        light deflection by a solar-system body
**
*/
```

```
void iauLdsun(double p[3], double e[3], double em, double p1[3])
/*
**  - - - - - - - - -
**   i a u L d s u n
**  - - - - - - - - -
**
**  Deflection of starlight by the Sun.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     p      double[3]  direction from observer to star (unit vector)
**     e      double[3]  direction from Sun to observer (unit vector)
**     em     double     distance from Sun to observer (au)
**
**  Returned:
**     p1     double[3]  observer to deflected star (unit vector)
**
**  Notes:
**
**  1) The source is presumed to be sufficiently distant that its
**     directions seen from the Sun and the observer are essentially
**     the same.
**
**  2) The deflection is restrained when the angle between the star and
**     the center of the Sun is less than a threshold value, falling to
**     zero deflection for zero separation.  The chosen threshold value
**     is within the solar limb for all solar-system applications, and
**     is about 5 arcminutes for the case of a terrestrial observer.
**
**  3) The arguments p and p1 can be the same array.
**
**  Called:
**     iauLd        light deflection by a solar-system body
**
*/
```

```
void iauLteceq(double epj, double dl, double db, double *dr, double *dd)
/*
**  - - - - - - - - - -
**   i a u L t e c e q
**  - - - - - - - - - -
**
**  Transformation from ecliptic coordinates (mean equinox and ecliptic
**  of date) to ICRS RA,Dec, using a long-term precession model.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     epj     double     Julian epoch (TT)
**     dl,db   double     ecliptic longitude and latitude (radians)
**
**  Returned:
**     dr,dd   double     ICRS right ascension and declination (radians)
**
**  1) No assumptions are made about whether the coordinates represent
**     starlight and embody astrometric effects such as parallax or
**     aberration.
**
**  2) The transformation is approximately that from ecliptic longitude
**     and latitude (mean equinox and ecliptic of date) to mean J2000.0
**     right ascension and declination, with only frame bias (always
**     less than 25 mas) to disturb this classical picture.
**
**  3) The Vondrak et al. (2011, 2012) 400 millennia precession model
**     agrees with the IAU 2006 precession at J2000.0 and stays within
**     100 microarcseconds during the 20th and 21st centuries.  It is
**     accurate to a few arcseconds throughout the historical period,
**     worsening to a few tenths of a degree at the end of the
**     +/- 200,000 year time span.
**
**  Called:
**     iauS2c       spherical coordinates to unit vector
**     iauLtecm     J2000.0 to ecliptic rotation matrix, long term
**     iauTrxp      product of transpose of r-matrix and p-vector
**     iauC2s       unit vector to spherical coordinates
**     iauAnp       normalize angle into range 0 to 2pi
**     iauAnpm      normalize angle into range +/- pi
**
**  References:
**
**    Vondrak, J., Capitaine, N. and Wallace, P., 2011, New precession
**    expressions, valid for long time intervals, Astron.Astrophys. 534,
**    A22
**
**    Vondrak, J., Capitaine, N. and Wallace, P., 2012, New precession
**    expressions, valid for long time intervals (Corrigendum),
**    Astron.Astrophys. 541, C1
**
*/
```

```
void iauLtecm(double epj, double rm[3][3])
/*
**  - - - - - - - - -
**   i a u L t e c m
**  - - - - - - - - -
**
**  ICRS equatorial to ecliptic rotation matrix, long-term.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     epj     double         Julian epoch (TT)
**
**  Returned:
**     rm      double[3][3]   ICRS to ecliptic rotation matrix
**
**  Notes:
**
**  1) The matrix is in the sense
**
**        E_ep = rm x P_ICRS,
**
**     where P_ICRS is a vector with respect to ICRS right ascension
**     and declination axes and E_ep is the same vector with respect to
**     the (inertial) ecliptic and equinox of epoch epj.
**
**  2) P_ICRS is a free vector, merely a direction, typically of unit
**     magnitude, and not bound to any particular spatial origin, such
**     as the Earth, Sun or SSB.  No assumptions are made about whether
**     it represents starlight and embodies astrometric effects such as
**     parallax or aberration.  The transformation is approximately that
**     between mean J2000.0 right ascension and declination and ecliptic
**     longitude and latitude, with only frame bias (always less than
**     25 mas) to disturb this classical picture.
**
**  3) The Vondrak et al. (2011, 2012) 400 millennia precession model
**     agrees with the IAU 2006 precession at J2000.0 and stays within
**     100 microarcseconds during the 20th and 21st centuries.  It is
**     accurate to a few arcseconds throughout the historical period,
**     worsening to a few tenths of a degree at the end of the
**     +/- 200,000 year time span.
**
**  Called:
**     iauLtpequ    equator pole, long term
**     iauLtpecl    ecliptic pole, long term
**     iauPxp       vector product
**     iauPn        normalize vector
**
**  References:
**
**    Vondrak, J., Capitaine, N. and Wallace, P., 2011, New precession
**    expressions, valid for long time intervals, Astron.Astrophys. 534,
**    A22
**
**    Vondrak, J., Capitaine, N. and Wallace, P., 2012, New precession
**    expressions, valid for long time intervals (Corrigendum),
**    Astron.Astrophys. 541, C1
**
*/
```

```
void iauLteqec(double epj, double dr, double dd, double *dl, double *db)
/*
**  - - - - - - - - - -
**   i a u L t e q e c
**  - - - - - - - - - -
**
**  Transformation from ICRS RA,Dec to ecliptic coordinates (mean equinox
**  and ecliptic of date), using a long-term precession model.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     epj     double     Julian epoch (TT)
**     dr,dd   double     ICRS right ascension and declination (radians)
**
**  Returned:
**     dl,db   double     ecliptic longitude and latitude (radians)
**
**  1) No assumptions are made about whether the coordinates represent
**     starlight and embody astrometric effects such as parallax or
**     aberration.
**
**  2) The transformation is approximately that from mean J2000.0 right
**     ascension and declination to ecliptic longitude and latitude
**     (mean equinox and ecliptic of date), with only frame bias (always
**     less than 25 mas) to disturb this classical picture.
**
**  3) The Vondrak et al. (2011, 2012) 400 millennia precession model
**     agrees with the IAU 2006 precession at J2000.0 and stays within
**     100 microarcseconds during the 20th and 21st centuries.  It is
**     accurate to a few arcseconds throughout the historical period,
**     worsening to a few tenths of a degree at the end of the
**     +/- 200,000 year time span.
**
**  Called:
**     iauS2c       spherical coordinates to unit vector
**     iauLtecm     J2000.0 to ecliptic rotation matrix, long term
**     iauRxp       product of r-matrix and p-vector
**     iauC2s       unit vector to spherical coordinates
**     iauAnp       normalize angle into range 0 to 2pi
**     iauAnpm      normalize angle into range +/- pi
**
**  References:
**
**    Vondrak, J., Capitaine, N. and Wallace, P., 2011, New precession
**    expressions, valid for long time intervals, Astron.Astrophys. 534,
**    A22
**
**    Vondrak, J., Capitaine, N. and Wallace, P., 2012, New precession
**    expressions, valid for long time intervals (Corrigendum),
**    Astron.Astrophys. 541, C1
**
*/
```

```
void iauLtp(double epj, double rp[3][3])
/*
**  - - - - - - -
**   i a u L t p
**  - - - - - - -
**
**  Long-term precession matrix.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     epj     double         Julian epoch (TT)
**
**  Returned:
**     rp      double[3][3]   precession matrix, J2000.0 to date
**
**  Notes:
**
**  1) The matrix is in the sense
**
**        P_date = rp x P_J2000,
**
**     where P_J2000 is a vector with respect to the J2000.0 mean
**     equator and equinox and P_date is the same vector with respect to
**     the mean equator and equinox of epoch epj.
**
**  2) The Vondrak et al. (2011, 2012) 400 millennia precession model
**     agrees with the IAU 2006 precession at J2000.0 and stays within
**     100 microarcseconds during the 20th and 21st centuries.  It is
**     accurate to a few arcseconds throughout the historical period,
**     worsening to a few tenths of a degree at the end of the
**     +/- 200,000 year time span.
**
**  Called:
**     iauLtpequ    equator pole, long term
**     iauLtpecl    ecliptic pole, long term
**     iauPxp       vector product
**     iauPn        normalize vector
**
**  References:
**
**    Vondrak, J., Capitaine, N. and Wallace, P., 2011, New precession
**    expressions, valid for long time intervals, Astron.Astrophys. 534,
**    A22
**
**    Vondrak, J., Capitaine, N. and Wallace, P., 2012, New precession
**    expressions, valid for long time intervals (Corrigendum),
**    Astron.Astrophys. 541, C1
**
*/
```

```
void iauLtpb(double epj, double rpb[3][3])
/*
**  - - - - - - - -
**   i a u L t p b
**  - - - - - - - -
**
**  Long-term precession matrix, including ICRS frame bias.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     epj     double         Julian epoch (TT)
**
**  Returned:
**     rpb     double[3][3]   precession+bias matrix, J2000.0 to date
**
**  Notes:
**
**  1) The matrix is in the sense
**
**        P_date = rpb x P_ICRS,
**
**     where P_ICRS is a vector in the Geocentric Celestial Reference
**     System, and P_date is the vector with respect to the Celestial
**     Intermediate Reference System at that date but with nutation
**     neglected.
**
**  2) A first order frame bias formulation is used, of sub-
**     microarcsecond accuracy compared with a full 3D rotation.
**
**  3) The Vondrak et al. (2011, 2012) 400 millennia precession model
**     agrees with the IAU 2006 precession at J2000.0 and stays within
**     100 microarcseconds during the 20th and 21st centuries.  It is
**     accurate to a few arcseconds throughout the historical period,
**     worsening to a few tenths of a degree at the end of the
**     +/- 200,000 year time span.
**
**  References:
**
**    Vondrak, J., Capitaine, N. and Wallace, P., 2011, New precession
**    expressions, valid for long time intervals, Astron.Astrophys. 534,
**    A22
**
**    Vondrak, J., Capitaine, N. and Wallace, P., 2012, New precession
**    expressions, valid for long time intervals (Corrigendum),
**    Astron.Astrophys. 541, C1
**
*/
```

```
void iauLtpecl(double epj, double vec[3])
/*
**  - - - - - - - - - -
**   i a u L t p e c l
**  - - - - - - - - - -
**
**  Long-term precession of the ecliptic.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     epj     double         Julian epoch (TT)
**
**  Returned:
**     vec     double[3]      ecliptic pole unit vector
**
**  Notes:
**
**  1) The returned vector is with respect to the J2000.0 mean equator
**     and equinox.
**
**  2) The Vondrak et al. (2011, 2012) 400 millennia precession model
**     agrees with the IAU 2006 precession at J2000.0 and stays within
**     100 microarcseconds during the 20th and 21st centuries.  It is
**     accurate to a few arcseconds throughout the historical period,
**     worsening to a few tenths of a degree at the end of the
**     +/- 200,000 year time span.
**
**  References:
**
**    Vondrak, J., Capitaine, N. and Wallace, P., 2011, New precession
**    expressions, valid for long time intervals, Astron.Astrophys. 534,
**    A22
**
**    Vondrak, J., Capitaine, N. and Wallace, P., 2012, New precession
**    expressions, valid for long time intervals (Corrigendum),
**    Astron.Astrophys. 541, C1
**
*/
```

```
void iauLtpequ(double epj, double veq[3])
/*
**  - - - - - - - - - -
**   i a u L t p e q u
**  - - - - - - - - - -
**
**  Long-term precession of the equator.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     epj     double         Julian epoch (TT)
**
**  Returned:
**     veq     double[3]      equator pole unit vector
**
**  Notes:
**
**  1) The returned vector is with respect to the J2000.0 mean equator
**     and equinox.
**
**  2) The Vondrak et al. (2011, 2012) 400 millennia precession model
**     agrees with the IAU 2006 precession at J2000.0 and stays within
**     100 microarcseconds during the 20th and 21st centuries.  It is
**     accurate to a few arcseconds throughout the historical period,
**     worsening to a few tenths of a degree at the end of the
**     +/- 200,000 year time span.
**
**  References:
**
**    Vondrak, J., Capitaine, N. and Wallace, P., 2011, New precession
**    expressions, valid for long time intervals, Astron.Astrophys. 534,
**    A22
**
**    Vondrak, J., Capitaine, N. and Wallace, P., 2012, New precession
**    expressions, valid for long time intervals (Corrigendum),
**    Astron.Astrophys. 541, C1
**
*/
```

```
void iauMoon98 ( double date1, double date2, double pv[2][3] )
/*
**  - - - - - - - - - -
**   i a u M o o n 9 8
**  - - - - - - - - - -
**
**  Approximate geocentric position and velocity of the Moon.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  n.b. Not IAU-endorsed and without canonical status.
**
**  Given:
**     date1  double         TT date part A (Notes 1,4)
**     date2  double         TT date part B (Notes 1,4)
**
**  Returned:
**     pv     double[2][3]   Moon p,v, GCRS (au, au/d, Note 5)
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways, among
**     others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in cases
**     where the loss of several decimal digits of resolution is
**     acceptable.  The J2000 method is best matched to the way the
**     argument is handled internally and will deliver the optimum
**     resolution.  The MJD method and the date & time methods are both
**     good compromises between resolution and convenience.  The limited
**     accuracy of the present algorithm is such that any of the methods
**     is satisfactory.
**
**  2) This function is a full implementation of the algorithm
**     published by Meeus (see reference) except that the light-time
**     correction to the Moon's mean longitude has been omitted.
**
**  3) Comparisons with ELP/MPP02 over the interval 1950-2100 gave RMS
**     errors of 2.9 arcsec in geocentric direction, 6.1 km in position
**     and 36 mm/s in velocity.  The worst case errors were 18.3 arcsec
**     in geocentric direction, 31.7 km in position and 172 mm/s in
**     velocity.
**
**  4) The original algorithm is expressed in terms of "dynamical time",
**     which can either be TDB or TT without any significant change in
**     accuracy.  UT cannot be used without incurring significant errors
**     (30 arcsec in the present era) due to the Moon's 0.5 arcsec/sec
**     movement.
**
**  5) The result is with respect to the GCRS (the same as J2000.0 mean
**     equator and equinox to within 23 mas).
**
**  6) Velocity is obtained by a complete analytical differentiation
**     of the Meeus model.
**
**  7) The Meeus algorithm generates position and velocity in mean
**     ecliptic coordinates of date, which the present function then
**     rotates into GCRS.  Because the ecliptic system is precessing,
**     there is a coupling between this spin (about 1.4 degrees per
```

```
**     century) and the Moon position that produces a small velocity
**     contribution.  In the present function this effect is neglected
**     as it corresponds to a maximum difference of less than 3 mm/s and
**     increases the RMS error by only 0.4%.
**
**  References:
**
**     Meeus, J., Astronomical Algorithms, 2nd edition, Willmann-Bell,
**     1998, p337.
**
**     Simon, J.L., Bretagnon, P., Chapront, J., Chapront-Touze, M.,
**     Francou, G. & Laskar, J., Astron.Astrophys., 1994, 282, 663
**
**  Defined in sofam.h:
**     DAU           astronomical unit (m)
**     DJC           days per Julian century
**     DJ00          reference epoch (J2000.0), Julian Date
**     DD2R          degrees to radians
**
**  Called:
**     iauS2pv      spherical coordinates to pv-vector
**     iauPfw06     bias-precession F-W angles, IAU 2006
**     iauIr        initialize r-matrix to identity
**     iauRz        rotate around Z-axis
**     iauRx        rotate around X-axis
**     iauRxpv      product of r-matrix and pv-vector
**
*/
```

```
void iauNum00a(double date1, double date2, double rmatn[3][3])
/*
**  - - - - - - - - - -
**   i a u N u m 0 0 a
**  - - - - - - - - - -
**
**  Form the matrix of nutation for a given date, IAU 2000A model.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1,date2  double          TT as a 2-part Julian Date (Note 1)
**
**  Returned:
**     rmatn        double[3][3]    nutation matrix
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The matrix operates in the sense V(true) = rmatn * V(mean), where
**     the p-vector V(true) is with respect to the true equatorial triad
**     of date and the p-vector V(mean) is with respect to the mean
**     equatorial triad of date.
**
**  3) A faster, but slightly less accurate, result (about 1 mas) can be
**     obtained by using instead the iauNum00b function.
**
**  Called:
**     iauPn00a     bias/precession/nutation, IAU 2000A
**
**  Reference:
**
**     Explanatory Supplement to the Astronomical Almanac,
**     P. Kenneth Seidelmann (ed), University Science Books (1992),
**     Section 3.222-3 (p114).
**
*/
```

```
void iauNum00b(double date1, double date2, double rmatn[3][3])
/*
**  - - - - - - - - - -
**   i a u N u m 0 0 b
**  - - - - - - - - - -
**
**  Form the matrix of nutation for a given date, IAU 2000B model.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1,date2  double         TT as a 2-part Julian Date (Note 1)
**
**  Returned:
**     rmatn        double[3][3]   nutation matrix
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The matrix operates in the sense V(true) = rmatn * V(mean), where
**     the p-vector V(true) is with respect to the true equatorial triad
**     of date and the p-vector V(mean) is with respect to the mean
**     equatorial triad of date.
**
**  3) The present function is faster, but slightly less accurate (about
**     1 mas), than the iauNum00a function.
**
**  Called:
**     iauPn00b     bias/precession/nutation, IAU 2000B
**
**  Reference:
**
**     Explanatory Supplement to the Astronomical Almanac,
**     P. Kenneth Seidelmann (ed), University Science Books (1992),
**     Section 3.222-3 (p114).
**
*/
```

```
void iauNum06a(double date1, double date2, double rmatn[3][3])
/*
**  - - - - - - - - - - -
**   i a u N u m 0 6 a
**  - - - - - - - - - - -
**
**  Form the matrix of nutation for a given date, IAU 2006/2000A model.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1,date2   double          TT as a 2-part Julian Date (Note 1)
**
**  Returned:
**     rmatn         double[3][3]    nutation matrix
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The matrix operates in the sense V(true) = rmatn * V(mean), where
**     the p-vector V(true) is with respect to the true equatorial triad
**     of date and the p-vector V(mean) is with respect to the mean
**     equatorial triad of date.
**
**  Called:
**     iauObl06     mean obliquity, IAU 2006
**     iauNut06a    nutation, IAU 2006/2000A
**     iauNumat     form nutation matrix
**
**  Reference:
**
**     Explanatory Supplement to the Astronomical Almanac,
**     P. Kenneth Seidelmann (ed), University Science Books (1992),
**     Section 3.222-3 (p114).
**
*/
```

```
void iauNumat(double epsa, double dpsi, double deps, double rmatn[3][3])
/*
**  - - - - - - - - - -
**   i a u N u m a t
**  - - - - - - - - - -
**
**  Form the matrix of nutation.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     epsa        double         mean obliquity of date (Note 1)
**     dpsi,deps   double         nutation (Note 2)
**
**  Returned:
**     rmatn       double[3][3]   nutation matrix (Note 3)
**
**  Notes:
**
**
**  1) The supplied mean obliquity epsa, must be consistent with the
**     precession-nutation models from which dpsi and deps were obtained.
**
**  2) The caller is responsible for providing the nutation components;
**     they are in longitude and obliquity, in radians and are with
**     respect to the equinox and ecliptic of date.
**
**  3) The matrix operates in the sense V(true) = rmatn * V(mean),
**     where the p-vector V(true) is with respect to the true
**     equatorial triad of date and the p-vector V(mean) is with
**     respect to the mean equatorial triad of date.
**
**  Called:
**     iauIr        initialize r-matrix to identity
**     iauRx        rotate around X-axis
**     iauRz        rotate around Z-axis
**
**  Reference:
**
**     Explanatory Supplement to the Astronomical Almanac,
**     P. Kenneth Seidelmann (ed), University Science Books (1992),
**     Section 3.222-3 (p114).
**
*/
```

```
void iauNut00a(double date1, double date2, double *dpsi, double *deps)
/*
**  - - - - - - - - - -
**   i a u N u t 0 0 a
**  - - - - - - - - - -
**
**  Nutation, IAU 2000A model (MHB2000 luni-solar and planetary nutation
**  with free core nutation omitted).
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical model.
**
**  Given:
**     date1,date2   double   TT as a 2-part Julian Date (Note 1)
**
**  Returned:
**     dpsi,deps     double   nutation, luni-solar + planetary (Note 2)
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The nutation components in longitude and obliquity are in radians
**     and with respect to the equinox and ecliptic of date.  The
**     obliquity at J2000.0 is assumed to be the Lieske et al. (1977)
**     value of 84381.448 arcsec.
**
**     Both the luni-solar and planetary nutations are included.  The
**     latter are due to direct planetary nutations and the
**     perturbations of the lunar and terrestrial orbits.
**
**  3) The function computes the MHB2000 nutation series with the
**     associated corrections for planetary nutations.  It is an
**     implementation of the nutation part of the IAU 2000A precession-
**     nutation model, formally adopted by the IAU General Assembly in
**     2000, namely MHB2000 (Mathews et al. 2002), but with the free
**     core nutation (FCN - see Note 4) omitted.
**
**  4) The full MHB2000 model also contains contributions to the
**     nutations in longitude and obliquity due to the free-excitation
**     of the free-core-nutation during the period 1979-2000.  These FCN
**     terms, which are time-dependent and unpredictable, are NOT
**     included in the present function and, if required, must be
**     independently computed.  With the FCN corrections included, the
**     present function delivers a pole which is at current epochs
**     accurate to a few hundred microarcseconds.  The omission of FCN
**     introduces further errors of about that size.
**
**  5) The present function provides classical nutation.  The MHB2000
**     algorithm, from which it is adapted, deals also with (i) the
**     offsets between the GCRS and mean poles and (ii) the adjustments
**     in longitude and obliquity due to the changed precession rates.
```

```
**     These additional functions, namely frame bias and precession
**     adjustments, are supported by the SOFA functions iauBi00  and
**     iauPr00.
**
**  6) The MHB2000 algorithm also provides "total" nutations, comprising
**     the arithmetic sum of the frame bias, precession adjustments,
**     luni-solar nutation and planetary nutation.  These total
**     nutations can be used in combination with an existing IAU 1976
**     precession implementation, such as iauPmat76,  to deliver GCRS-
**     to-true predictions of sub-mas accuracy at current dates.
**     However, there are three shortcomings in the MHB2000 model that
**     must be taken into account if more accurate or definitive results
**     are required (see Wallace 2002):
**
**       (i) The MHB2000 total nutations are simply arithmetic sums,
**           yet in reality the various components are successive Euler
**           rotations.  This slight lack of rigor leads to cross terms
**           that exceed 1 mas after a century.  The rigorous procedure
**           is to form the GCRS-to-true rotation matrix by applying the
**           bias, precession and nutation in that order.
**
**      (ii) Although the precession adjustments are stated to be with
**           respect to Lieske et al. (1977), the MHB2000 model does
**           not specify which set of Euler angles are to be used and
**           how the adjustments are to be applied.  The most literal
**           and straightforward procedure is to adopt the 4-rotation
**           epsilon_0, psi_A, omega_A, xi_A option, and to add DPSIPR
**           to psi_A and DEPSPR to both omega_A and eps_A.
**
**     (iii) The MHB2000 model predates the determination by Chapront
**           et al. (2002) of a 14.6 mas displacement between the
**           J2000.0 mean equinox and the origin of the ICRS frame.  It
**           should, however, be noted that neglecting this displacement
**           when calculating star coordinates does not lead to a
**           14.6 mas change in right ascension, only a small second-
**           order distortion in the pattern of the precession-nutation
**           effect.
**
**     For these reasons, the SOFA functions do not generate the "total
**     nutations" directly, though they can of course easily be
**     generated by calling iauBi00, iauPr00 and the present function
**     and adding the results.
**
**  7) The MHB2000 model contains 41 instances where the same frequency
**     appears multiple times, of which 38 are duplicates and three are
**     triplicates.  To keep the present code close to the original MHB
**     algorithm, this small inefficiency has not been corrected.
**
**  Called:
**     iauFal03     mean anomaly of the Moon
**     iauFaf03     mean argument of the latitude of the Moon
**     iauFaom03    mean longitude of the Moon's ascending node
**     iauFame03    mean longitude of Mercury
**     iauFave03    mean longitude of Venus
**     iauFae03     mean longitude of Earth
**     iauFama03    mean longitude of Mars
**     iauFaju03    mean longitude of Jupiter
**     iauFasa03    mean longitude of Saturn
**     iauFaur03    mean longitude of Uranus
**     iauFapa03    general accumulated precession in longitude
**
**  References:
**
**     Chapront, J., Chapront-Touze, M. & Francou, G. 2002,
**     Astron.Astrophys. 387, 700
**
**     Lieske, J.H., Lederle, T., Fricke, W. & Morando, B. 1977,
**     Astron.Astrophys. 58, 1-16
**
**     Mathews, P.M., Herring, T.A., Buffet, B.A. 2002, J.Geophys.Res.
**     107, B4.  The MHB_2000 code itself was obtained on 9th September
**     2002 from ftp//maia.usno.navy.mil/conv2000/chapter5/IAU2000A.
**
```

```
**      Simon, J.-L., Bretagnon, P., Chapront, J., Chapront-Touze, M.,
**      Francou, G., Laskar, J. 1994, Astron.Astrophys. 282, 663-683
**
**      Souchay, J., Loysel, B., Kinoshita, H., Folgueira, M. 1999,
**      Astron.Astrophys.Supp.Ser. 135, 111
**
**      Wallace, P.T., "Software for Implementing the IAU 2000
**      Resolutions", in IERS Workshop 5.1 (2002)
**
*/
```

```
void iauNut00b(double date1, double date2, double *dpsi, double *deps)
/*
**  - - - - - - - - - -
**   i a u N u t 0 0 b
**  - - - - - - - - - -
**
**  Nutation, IAU 2000B model.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical model.
**
**  Given:
**     date1,date2   double    TT as a 2-part Julian Date (Note 1)
**
**  Returned:
**     dpsi,deps     double    nutation, luni-solar + planetary (Note 2)
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The nutation components in longitude and obliquity are in radians
**     and with respect to the equinox and ecliptic of date.  The
**     obliquity at J2000.0 is assumed to be the Lieske et al. (1977)
**     value of 84381.448 arcsec.  (The errors that result from using
**     this function with the IAU 2006 value of 84381.406 arcsec can be
**     neglected.)
**
**     The nutation model consists only of luni-solar terms, but
**     includes also a fixed offset which compensates for certain long-
**     period planetary terms (Note 7).
**
**  3) This function is an implementation of the IAU 2000B abridged
**     nutation model formally adopted by the IAU General Assembly in
**     2000.  The function computes the MHB_2000_SHORT luni-solar
**     nutation series (Luzum 2001), but without the associated
**     corrections for the precession rate adjustments and the offset
**     between the GCRS and J2000.0 mean poles.
**
**  4) The full IAU 2000A (MHB2000) nutation model contains nearly 1400
**     terms.  The IAU 2000B model (McCarthy & Luzum 2003) contains only
**     77 terms, plus additional simplifications, yet still delivers
**     results of 1 mas accuracy at present epochs.  This combination of
**     accuracy and size makes the IAU 2000B abridged nutation model
**     suitable for most practical applications.
**
**     The function delivers a pole accurate to 1 mas from 1900 to 2100
**     (usually better than 1 mas, very occasionally just outside
**     1 mas).  The full IAU 2000A model, which is implemented in the
**     function iauNut00a (q.v.), delivers considerably greater accuracy
**     at current dates;  however, to realize this improved accuracy,
**     corrections for the essentially unpredictable free-core-nutation
```

```
**     (FCN) must also be included.
**
**  5) The present function provides classical nutation.  The
**     MHB_2000_SHORT algorithm, from which it is adapted, deals also
**     with (i) the offsets between the GCRS and mean poles and (ii) the
**     adjustments in longitude and obliquity due to the changed
**     precession rates.  These additional functions, namely frame bias
**     and precession adjustments, are supported by the SOFA functions
**     iauBi00  and iauPr00.
**
**  6) The MHB_2000_SHORT algorithm also provides "total" nutations,
**     comprising the arithmetic sum of the frame bias, precession
**     adjustments, and nutation (luni-solar + planetary).  These total
**     nutations can be used in combination with an existing IAU 1976
**     precession implementation, such as iauPmat76,  to deliver GCRS-
**     to-true predictions of mas accuracy at current epochs.  However,
**     for symmetry with the iauNut00a  function (q.v. for the reasons),
**     the SOFA functions do not generate the "total nutations"
**     directly.  Should they be required, they could of course easily
**     be generated by calling iauBi00, iauPr00 and the present function
**     and adding the results.
**
**  7) The IAU 2000B model includes "planetary bias" terms that are
**     fixed in size but compensate for long-period nutations.  The
**     amplitudes quoted in McCarthy & Luzum (2003), namely
**     Dpsi = -1.5835 mas and Depsilon = +1.6339 mas, are optimized for
**     the "total nutations" method described in Note 6.  The Luzum
**     (2001) values used in this SOFA implementation, namely -0.135 mas
**     and +0.388 mas, are optimized for the "rigorous" method, where
**     frame bias, precession and nutation are applied separately and in
**     that order.  During the interval 1995-2050, the SOFA
**     implementation delivers a maximum error of 1.001 mas (not
**     including FCN).
**
**  References:
**
**     Lieske, J.H., Lederle, T., Fricke, W., Morando, B., "Expressions
**     for the precession quantities based upon the IAU /1976/ system of
**     astronomical constants", Astron.Astrophys. 58, 1-2, 1-16. (1977)
**
**     Luzum, B., private communication, 2001 (Fortran code
**     MHB_2000_SHORT)
**
**     McCarthy, D.D. & Luzum, B.J., "An abridged model of the
**     precession-nutation of the celestial pole", Cel.Mech.Dyn.Astron.
**     85, 37-49 (2003)
**
**     Simon, J.-L., Bretagnon, P., Chapront, J., Chapront-Touze, M.,
**     Francou, G., Laskar, J., Astron.Astrophys. 282, 663-683 (1994)
**
*/
```

```
void iauNut06a(double date1, double date2, double *dpsi, double *deps)
/*
**  - - - - - - - - - -
**   i a u N u t 0 6 a
**  - - - - - - - - - -
**
**  IAU 2000A nutation with adjustments to match the IAU 2006
**  precession.
**
**  Given:
**     date1,date2   double   TT as a 2-part Julian Date (Note 1)
**
**  Returned:
**     dpsi,deps     double   nutation, luni-solar + planetary (Note 2)
**
**  Status:  canonical model.
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The nutation components in longitude and obliquity are in radians
**     and with respect to the mean equinox and ecliptic of date,
**     IAU 2006 precession model (Hilton et al. 2006, Capitaine et al.
**     2005).
**
**  3) The function first computes the IAU 2000A nutation, then applies
**     adjustments for (i) the consequences of the change in obliquity
**     from the IAU 1980 ecliptic to the IAU 2006 ecliptic and (ii) the
**     secular variation in the Earth's dynamical form factor J2.
**
**  4) The present function provides classical nutation, complementing
**     the IAU 2000 frame bias and IAU 2006 precession.  It delivers a
**     pole which is at current epochs accurate to a few tens of
**     microarcseconds, apart from the free core nutation.
**
**  Called:
**     iauNut00a    nutation, IAU 2000A
**
**  References:
**
**     Chapront, J., Chapront-Touze, M. & Francou, G. 2002,
**     Astron.Astrophys. 387, 700
**
**     Lieske, J.H., Lederle, T., Fricke, W. & Morando, B. 1977,
**     Astron.Astrophys. 58, 1-16
**
**     Mathews, P.M., Herring, T.A., Buffet, B.A. 2002, J.Geophys.Res.
**     107, B4.  The MHB_2000 code itself was obtained on 9th September
**     2002 from ftp//maia.usno.navy.mil/conv2000/chapter5/IAU2000A.
**
**     Simon, J.-L., Bretagnon, P., Chapront, J., Chapront-Touze, M.,
**     Francou, G., Laskar, J. 1994, Astron.Astrophys. 282, 663-683
**
```

```
**      Souchay, J., Loysel, B., Kinoshita, H., Folgueira, M. 1999,
**      Astron.Astrophys.Supp.Ser. 135, 111
**
**      Wallace, P.T., "Software for Implementing the IAU 2000
**      Resolutions", in IERS Workshop 5.1 (2002)
**
*/
```

```
void iauNut80(double date1, double date2, double *dpsi, double *deps)
/*
**  - - - - - - - - -
**   i a u N u t 8 0
**  - - - - - - - - -
**
**  Nutation, IAU 1980 model.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical model.
**
**  Given:
**     date1,date2   double    TT as a 2-part Julian Date (Note 1)
**
**  Returned:
**     dpsi          double    nutation in longitude (radians)
**     deps          double    nutation in obliquity (radians)
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The nutation components are with respect to the ecliptic of
**     date.
**
**  Called:
**     iauAnpm      normalize angle into range +/- pi
**
**  Reference:
**
**     Explanatory Supplement to the Astronomical Almanac,
**     P. Kenneth Seidelmann (ed), University Science Books (1992),
**     Section 3.222 (p111).
**
*/
```

```
void iauNutm80(double date1, double date2, double rmatn[3][3])
/*
**  - - - - - - - - - -
**   i a u N u t m 8 0
**  - - - - - - - - - -
**
**  Form the matrix of nutation for a given date, IAU 1980 model.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1,date2    double          TDB date (Note 1)
**
**  Returned:
**     rmatn          double[3][3]    nutation matrix
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The matrix operates in the sense V(true) = rmatn * V(mean),
**     where the p-vector V(true) is with respect to the true
**     equatorial triad of date and the p-vector V(mean) is with
**     respect to the mean equatorial triad of date.
**
**  Called:
**     iauNut80     nutation, IAU 1980
**     iauObl80     mean obliquity, IAU 1980
**     iauNumat     form nutation matrix
**
*/
```

```
double iauObl06(double date1, double date2)
/*
**  - - - - - - - - -
**   i a u O b l 0 6
**  - - - - - - - - -
**
**  Mean obliquity of the ecliptic, IAU 2006 precession model.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical model.
**
**  Given:
**     date1,date2  double   TT as a 2-part Julian Date (Note 1)
**
**  Returned (function value):
**                  double   obliquity of the ecliptic (radians, Note 2)
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The result is the angle between the ecliptic and mean equator of
**     date date1+date2.
**
**  Reference:
**
**     Hilton, J. et al., 2006, Celest.Mech.Dyn.Astron. 94, 351
**
*/
```

```
double iauObl80(double date1, double date2)
/*
**  - - - - - - - - -
**   i a u O b l 8 0
**  - - - - - - - - -
**
**  Mean obliquity of the ecliptic, IAU 1980 model.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical model.
**
**  Given:
**     date1,date2   double    TT as a 2-part Julian Date (Note 1)
**
**  Returned (function value):
**                   double    obliquity of the ecliptic (radians, Note 2)
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The result is the angle between the ecliptic and mean equator of
**     date date1+date2.
**
**  Reference:
**
**     Explanatory Supplement to the Astronomical Almanac,
**     P. Kenneth Seidelmann (ed), University Science Books (1992),
**     Expression 3.222-1 (p114).
**
*/
```

```
void iauP06e(double date1, double date2,
             double *eps0, double *psia, double *oma, double *bpa,
             double *bqa, double *pia, double *bpia,
             double *epsa, double *chia, double *za, double *zetaa,
             double *thetaa, double *pa,
             double *gam, double *phi, double *psi)
/*
**  - - - - - - - -
**   i a u P 0 6 e
**  - - - - - - - -
**
**  Precession angles, IAU 2006, equinox based.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical models.
**
**  Given:
**     date1,date2   double   TT as a 2-part Julian Date (Note 1)
**
**  Returned (see Note 2):
**     eps0          double   epsilon_0
**     psia          double   psi_A
**     oma           double   omega_A
**     bpa           double   P_A
**     bqa           double   Q_A
**     pia           double   pi_A
**     bpia          double   Pi_A
**     epsa          double   obliquity epsilon_A
**     chia          double   chi_A
**     za            double   z_A
**     zetaa         double   zeta_A
**     thetaa        double   theta_A
**     pa            double   p_A
**     gam           double   F-W angle gamma_J2000
**     phi           double   F-W angle phi_J2000
**     psi           double   F-W angle psi_J2000
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) This function returns the set of equinox based angles for the
**     Capitaine et al. "P03" precession theory, adopted by the IAU in
**     2006.  The angles are set out in Table 1 of Hilton et al. (2006):
**
**     eps0   epsilon_0   obliquity at J2000.0
**     psia   psi_A       luni-solar precession
**     oma    omega_A     inclination of equator wrt J2000.0 ecliptic
**     bpa    P_A         ecliptic pole x, J2000.0 ecliptic triad
**     bqa    Q_A         ecliptic pole -y, J2000.0 ecliptic triad
**     pia    pi_A        angle between moving and J2000.0 ecliptics
**     bpia   Pi_A        longitude of ascending node of the ecliptic
```

```
**     epsa   epsilon_A   obliquity of the ecliptic
**     chia   chi_A       planetary precession
**     za     z_A         equatorial precession: -3rd 323 Euler angle
**     zetaa  zeta_A      equatorial precession: -1st 323 Euler angle
**     thetaa theta_A     equatorial precession: 2nd 323 Euler angle
**     pa     p_A         general precession (n.b. see below)
**     gam    gamma_J2000 J2000.0 RA difference of ecliptic poles
**     phi    phi_J2000   J2000.0 codeclination of ecliptic pole
**     psi    psi_J2000   longitude difference of equator poles, J2000.0
**
**     The returned values are all radians.
**
**     Note that the t^5 coefficient in the series for p_A from
**     Capitaine et al. (2003) is incorrectly signed in Hilton et al.
**     (2006).
**
**  3) Hilton et al. (2006) Table 1 also contains angles that depend on
**     models distinct from the P03 precession theory itself, namely the
**     IAU 2000A frame bias and nutation.  The quoted polynomials are
**     used in other SOFA functions:
**
**     . iauXy06  contains the polynomial parts of the X and Y series.
**
**     . iauS06  contains the polynomial part of the s+XY/2 series.
**
**     . iauPfw06  implements the series for the Fukushima-Williams
**       angles that are with respect to the GCRS pole (i.e. the variants
**       that include frame bias).
**
**  4) The IAU resolution stipulated that the choice of parameterization
**     was left to the user, and so an IAU compliant precession
**     implementation can be constructed using various combinations of
**     the angles returned by the present function.
**
**  5) The parameterization used by SOFA is the version of the Fukushima-
**     Williams angles that refers directly to the GCRS pole.  These
**     angles may be calculated by calling the function iauPfw06.  SOFA
**     also supports the direct computation of the CIP GCRS X,Y by
**     series, available by calling iauXy06.
**
**  6) The agreement between the different parameterizations is at the
**     1 microarcsecond level in the present era.
**
**  7) When constructing a precession formulation that refers to the GCRS
**     pole rather than the dynamical pole, it may (depending on the
**     choice of angles) be necessary to introduce the frame bias
**     explicitly.
**
**  8) It is permissible to re-use the same variable in the returned
**     arguments.  The quantities are stored in the stated order.
**
**  References:
**
**     Capitaine, N., Wallace, P.T. & Chapront, J., 2003,
**     Astron.Astrophys., 412, 567
**
**     Hilton, J. et al., 2006, Celest.Mech.Dyn.Astron. 94, 351
**
**  Called:
**     iauObl06     mean obliquity, IAU 2006
**
*/
```

```
void iauP2pv(double p[3], double pv[2][3])
/*
**  - - - - - - - - 
**   i a u P 2 p v
**  - - - - - - - - 
**
**  Extend a p-vector to a pv-vector by appending a zero velocity.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     p        double[3]       p-vector
**
**  Returned:
**     pv       double[2][3]    pv-vector
**
**  Called:
**     iauCp        copy p-vector
**     iauZp        zero p-vector
**
*/
```

```
void iauP2s(double p[3], double *theta, double *phi, double *r)
/*
**  - - - - - - -
**   i a u P 2 s
**  - - - - - - -
**
**  P-vector to spherical polar coordinates.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     p        double[3]    p-vector
**
**  Returned:
**     theta    double       longitude angle (radians)
**     phi      double       latitude angle (radians)
**     r        double       radial distance
**
**  Notes:
**
**  1) If P is null, zero theta, phi and r are returned.
**
**  2) At either pole, zero theta is returned.
**
**  Called:
**     iauC2s       p-vector to spherical
**     iauPm        modulus of p-vector
**
*/
```

```
double iauPap(double a[3], double b[3])
/*
**  - - - - - - -
**   i a u P a p
**  - - - - - - -
**
**  Position-angle from two p-vectors.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     a      double[3]  direction of reference point
**     b      double[3]  direction of point whose PA is required
**
**  Returned (function value):
**            double     position angle of b with respect to a (radians)
**
**  Notes:
**
**  1) The result is the position angle, in radians, of direction b with
**     respect to direction a.  It is in the range -pi to +pi.  The
**     sense is such that if b is a small distance "north" of a the
**     position angle is approximately zero, and if b is a small
**     distance "east" of a the position angle is approximately +pi/2.
**
**  2) The vectors a and b need not be of unit length.
**
**  3) Zero is returned if the two directions are the same or if either
**     vector is null.
**
**  4) If vector a is at a pole, the result is ill-defined.
**
**  Called:
**     iauPn        decompose p-vector into modulus and direction
**     iauPm        modulus of p-vector
**     iauPxp       vector product of two p-vectors
**     iauPmp       p-vector minus p-vector
**     iauPdp       scalar product of two p-vectors
**
*/
```

```
double iauPas(double al, double ap, double bl, double bp)
/*
**  - - - - - - -
**   i a u P a s
**  - - - - - - -
**
**  Position-angle from spherical coordinates.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     al     double     longitude of point A (e.g. RA) in radians
**     ap     double     latitude of point A (e.g. Dec) in radians
**     bl     double     longitude of point B
**     bp     double     latitude of point B
**
**  Returned (function value):
**            double     position angle of B with respect to A
**
**  Notes:
**
**  1) The result is the bearing (position angle), in radians, of point
**     B with respect to point A.  It is in the range -pi to +pi.  The
**     sense is such that if B is a small distance "east" of point A,
**     the bearing is approximately +pi/2.
**
**  2) Zero is returned if the two points are coincident.
**
*/
```

```
void iauPb06(double date1, double date2,
             double *bzeta, double *bz, double *btheta)
/*
**  - - - - - - - -
**   i a u P b 0 6
**  - - - - - - - -
**
**  This function forms three Euler angles which implement general
**  precession from epoch J2000.0, using the IAU 2006 model.  Frame
**  bias (the offset between ICRS and mean J2000.0) is included.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1,date2  double   TT as a 2-part Julian Date (Note 1)
**
**  Returned:
**     bzeta        double   1st rotation: radians cw around z
**     bz           double   3rd rotation: radians cw around z
**     btheta       double   2nd rotation: radians ccw around y
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The traditional accumulated precession angles zeta_A, z_A,
**     theta_A cannot be obtained in the usual way, namely through
**     polynomial expressions, because of the frame bias.  The latter
**     means that two of the angles undergo rapid changes near this
**     date.  They are instead the results of decomposing the
**     precession-bias matrix obtained by using the Fukushima-Williams
**     method, which does not suffer from the problem.  The
**     decomposition returns values which can be used in the
**     conventional formulation and which include frame bias.
**
**  3) The three angles are returned in the conventional order, which
**     is not the same as the order of the corresponding Euler
**     rotations.  The precession-bias matrix is
**     R_3(-z) x R_2(+theta) x R_3(-zeta).
**
**  4) Should zeta_A, z_A, theta_A angles be required that do not
**     contain frame bias, they are available by calling the SOFA
**     function iauP06e.
**
**  Called:
**     iauPmat06    PB matrix, IAU 2006
**     iauRz        rotate around Z-axis
**
*/
```

```
double iauPdp(double a[3], double b[3])
/*
**  - - - - - - -
**   i a u P d p
**  - - - - - - -
**
**  p-vector inner (=scalar=dot) product.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     a      double[3]     first p-vector
**     b      double[3]     second p-vector
**
**  Returned (function value):
**            double        a . b
**
*/
```

```
void iauPfw06(double date1, double date2,
              double *gamb, double *phib, double *psib, double *epsa)
/*
**  - - - - - - - - -
**   i a u P f w 0 6
**  - - - - - - - - -
**
**  Precession angles, IAU 2006 (Fukushima-Williams 4-angle formulation).
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical model.
**
**  Given:
**     date1,date2  double   TT as a 2-part Julian Date (Note 1)
**
**  Returned:
**     gamb         double   F-W angle gamma_bar (radians)
**     phib         double   F-W angle phi_bar (radians)
**     psib         double   F-W angle psi_bar (radians)
**     epsa         double   F-W angle epsilon_A (radians)
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) Naming the following points:
**
**           e = J2000.0 ecliptic pole,
**           p = GCRS pole,
**           E = mean ecliptic pole of date,
**     and   P = mean pole of date,
**
**     the four Fukushima-Williams angles are as follows:
**
**        gamb = gamma_bar = epE
**        phib = phi_bar = pE
**        psib = psi_bar = pEP
**        epsa = epsilon_A = EP
**
**  3) The matrix representing the combined effects of frame bias and
**     precession is:
**
**        PxB = R_1(-epsa).R_3(-psib).R_1(phib).R_3(gamb)
**
**  4) The matrix representing the combined effects of frame bias,
**     precession and nutation is simply:
**
**        NxPxB = R_1(-epsa-dE).R_3(-psib-dP).R_1(phib).R_3(gamb)
**
**     where dP and dE are the nutation components with respect to the
**     ecliptic of date.
**
```

```
**  Reference:
**
**     Hilton, J. et al., 2006, Celest.Mech.Dyn.Astron. 94, 351
**
**  Called:
**     iauObl06     mean obliquity, IAU 2006
**
*/
```

```
int iauPlan94(double date1, double date2, int np, double pv[2][3])
/*
**  - - - - - - - - - -
**   i a u P l a n 9 4
**  - - - - - - - - - -
**
**  Approximate heliocentric position and velocity of a nominated
**  planet:  Mercury, Venus, EMB, Mars, Jupiter, Saturn, Uranus or
**  Neptune (but not the Earth itself).
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  n.b. Not IAU-endorsed and without canonical status.
**
**  Given:
**     date1  double       TDB date part A (Note 1)
**     date2  double       TDB date part B (Note 1)
**     np     int          planet (1=Mercury, 2=Venus, 3=EMB, 4=Mars,
**                             5=Jupiter, 6=Saturn, 7=Uranus, 8=Neptune)
**
**  Returned (argument):
**     pv     double[2][3] planet p,v (heliocentric, J2000.0, au,au/d)
**
**  Returned (function value):
**            int          status: -1 = illegal NP (outside 1-8)
**                                  0 = OK
**                                 +1 = warning: year outside 1000-3000
**                                 +2 = warning: failed to converge
**
**  Notes:
**
**  1) The date date1+date2 is in the TDB time scale (in practice TT can
**     be used) and is a Julian Date, apportioned in any convenient way
**     between the two arguments.  For example, JD(TDB)=2450123.7 could
**     be expressed in any of these ways, among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in cases
**     where the loss of several decimal digits of resolution is
**     acceptable.  The J2000 method is best matched to the way the
**     argument is handled internally and will deliver the optimum
**     resolution.  The MJD method and the date & time methods are both
**     good compromises between resolution and convenience.  The limited
**     accuracy of the present algorithm is such that any of the methods
**     is satisfactory.
**
**  2) If an np value outside the range 1-8 is supplied, an error status
**     (function value -1) is returned and the pv vector set to zeroes.
**
**  3) For np=3 the result is for the Earth-Moon barycenter (EMB).  To
**     obtain the heliocentric position and velocity of the Earth, use
**     instead the SOFA function iauEpv00.
**
**  4) On successful return, the array pv contains the following:
**
**        pv[0][0]   x      }
**        pv[0][1]   y      } heliocentric position, au
**        pv[0][2]   z      }
**
**        pv[1][0]   xdot   }
**        pv[1][1]   ydot   } heliocentric velocity, au/d
**        pv[1][2]   zdot   }
```

```
**
**     The reference frame is equatorial and is with respect to the
**     mean equator and equinox of epoch J2000.0.
**
**  5) The algorithm is due to J.L. Simon, P. Bretagnon, J. Chapront,
**     M. Chapront-Touze, G. Francou and J. Laskar (Bureau des
**     Longitudes, Paris, France).  From comparisons with JPL
**     ephemeris DE102, they quote the following maximum errors
**     over the interval 1800-2050:
**
**                     L (arcsec)    B (arcsec)      R (km)
**
**        Mercury          4             1             300
**        Venus            5             1             800
**        EMB              6             1            1000
**        Mars            17             1            7700
**        Jupiter         71             5           76000
**        Saturn          81            13          267000
**        Uranus          86             7          712000
**        Neptune         11             1          253000
**
**     Over the interval 1000-3000, they report that the accuracy is no
**     worse than 1.5 times that over 1800-2050.  Outside 1000-3000 the
**     accuracy declines.
**
**     Comparisons of the present function with the JPL DE200 ephemeris
**     give the following RMS errors over the interval 1960-2025:
**
**                      position (km)     velocity (m/s)
**
**        Mercury            334               0.437
**        Venus             1060               0.855
**        EMB               2010               0.815
**        Mars              7690               1.98
**        Jupiter          71700               7.70
**        Saturn          199000              19.4
**        Uranus          564000              16.4
**        Neptune         158000              14.4
**
**     Comparisons against DE200 over the interval 1800-2100 gave the
**     following maximum absolute differences (the results using
**     DE406 were essentially the same):
**
**                   L (arcsec)   B (arcsec)     R (km)   Rdot (m/s)
**
**        Mercury        7            1            500       0.7
**        Venus          7            1           1100       0.9
**        EMB            9            1           1300       1.0
**        Mars          26            1           9000       2.5
**        Jupiter       78            6          82000       8.2
**        Saturn        87           14         263000      24.6
**        Uranus        86            7         661000      27.4
**        Neptune       11            2         248000      21.4
**
**  6) The present SOFA re-implementation of the original Simon et al.
**     Fortran code differs from the original in the following respects:
**
**       *  C instead of Fortran.
**
**       *  The date is supplied in two parts.
**
**       *  The result is returned only in equatorial Cartesian form;
**          the ecliptic longitude, latitude and radius vector are not
**          returned.
**
**       *  The result is in the J2000.0 equatorial frame, not ecliptic.
**
**       *  More is done in-line: there are fewer calls to subroutines.
**
**       *  Different error/warning status values are used.
**
**       *  A different Kepler's-equation-solver is used (avoiding
**          use of double precision complex).
```

```
**
**       *  Polynomials in t are nested to minimize rounding errors.
**
**       *  Explicit double constants are used to avoid mixed-mode
**          expressions.
**
**     None of the above changes affects the result significantly.
**
**  7) The returned status indicates the most serious condition
**     encountered during execution of the function.  Illegal np is
**     considered the most serious, overriding failure to converge,
**     which in turn takes precedence over the remote date warning.
**
**  Called:
**     iauAnpm      normalize angle into range +/- pi
**
**  Reference:  Simon, J.L, Bretagnon, P., Chapront, J.,
**              Chapront-Touze, M., Francou, G., and Laskar, J.,
**              Astron.Astrophys., 282, 663 (1994).
**
*/
```

```
double iauPm(double p[3])
/*
**  - - - - - -
**   i a u P m
**  - - - - - -
**
**  Modulus of p-vector.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     p      double[3]     p-vector
**
**  Returned (function value):
**            double        modulus
**
*/
```

```
void iauPmat00(double date1, double date2, double rbp[3][3])
/*
**  - - - - - - - - - -
**   i a u P m a t 0 0
**  - - - - - - - - - -
**
**  Precession matrix (including frame bias) from GCRS to a specified
**  date, IAU 2000 model.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1,date2  double          TT as a 2-part Julian Date (Note 1)
**
**  Returned:
**     rbp          double[3][3]    bias-precession matrix (Note 2)
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The matrix operates in the sense V(date) = rbp * V(GCRS), where
**     the p-vector V(GCRS) is with respect to the Geocentric Celestial
**     Reference System (IAU, 2000) and the p-vector V(date) is with
**     respect to the mean equatorial triad of the given date.
**
**  Called:
**     iauBp00      frame bias and precession matrices, IAU 2000
**
**  Reference:
**
**     IAU: Trans. International Astronomical Union, Vol. XXIVB;  Proc.
**     24th General Assembly, Manchester, UK.  Resolutions B1.3, B1.6.
**     (2000)
**
*/
```

```
void iauPmat06(double date1, double date2, double rbp[3][3])
/*
**  - - - - - - - - - -
**   i a u P m a t 0 6
**  - - - - - - - - - -
**
**  Precession matrix (including frame bias) from GCRS to a specified
**  date, IAU 2006 model.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1,date2  double          TT as a 2-part Julian Date (Note 1)
**
**  Returned:
**     rbp          double[3][3]    bias-precession matrix (Note 2)
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The matrix operates in the sense V(date) = rbp * V(GCRS), where
**     the p-vector V(GCRS) is with respect to the Geocentric Celestial
**     Reference System (IAU, 2000) and the p-vector V(date) is with
**     respect to the mean equatorial triad of the given date.
**
**  Called:
**     iauPfw06     bias-precession F-W angles, IAU 2006
**     iauFw2m      F-W angles to r-matrix
**
**  References:
**
**     Capitaine, N. & Wallace, P.T., 2006, Astron.Astrophys. 450, 855
**
**     IAU: Trans. International Astronomical Union, Vol. XXIVB;  Proc.
**     24th General Assembly, Manchester, UK.  Resolutions B1.3, B1.6.
**     (2000)
**
**     Wallace, P.T. & Capitaine, N., 2006, Astron.Astrophys. 459, 981
**
*/
```

```
void iauPmat76(double date1, double date2, double rmatp[3][3])
/*
**  - - - - - - - - - -
**   i a u P m a t 7 6
**  - - - - - - - - - -
**
**  Precession matrix from J2000.0 to a specified date, IAU 1976 model.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1,date2 double       ending date, TT (Note 1)
**
**  Returned:
**     rmatp       double[3][3] precession matrix, J2000.0 -> date1+date2
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The matrix operates in the sense V(date) = RMATP * V(J2000),
**     where the p-vector V(J2000) is with respect to the mean
**     equatorial triad of epoch J2000.0 and the p-vector V(date)
**     is with respect to the mean equatorial triad of the given
**     date.
**
**  3) Though the matrix method itself is rigorous, the precession
**     angles are expressed through canonical polynomials which are
**     valid only for a limited time span.  In addition, the IAU 1976
**     precession rate is known to be imperfect.  The absolute accuracy
**     of the present formulation is better than 0.1 arcsec from
**     1960AD to 2040AD, better than 1 arcsec from 1640AD to 2360AD,
**     and remains below 3 arcsec for the whole of the period
**     500BC to 3000AD.  The errors exceed 10 arcsec outside the
**     range 1200BC to 3900AD, exceed 100 arcsec outside 4200BC to
**     5600AD and exceed 1000 arcsec outside 6800BC to 8200AD.
**
**  Called:
**     iauPrec76    accumulated precession angles, IAU 1976
**     iauIr        initialize r-matrix to identity
**     iauRz        rotate around Z-axis
**     iauRy        rotate around Y-axis
**     iauCr        copy r-matrix
**
**  References:
**
**     Lieske, J.H., 1979, Astron.Astrophys. 73, 282.
**      equations (6) & (7), p283.
**
**     Kaplan,G.H., 1981. USNO circular no. 163, pA2.
**
```

```
*/
```

```
void iauPmp(double a[3], double b[3], double amb[3])
/*
**  - - - - - - -
**   i a u P m p
**  - - - - - - -
**
**  P-vector subtraction.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     a        double[3]      first p-vector
**     b        double[3]      second p-vector
**
**  Returned:
**     amb      double[3]      a - b
**
**  Note:
**     It is permissible to re-use the same array for any of the
**     arguments.
**
*/
```

```
void iauPmpx(double rc, double dc, double pr, double pd,
             double px, double rv, double pmt, double pob[3],
             double pco[3])
/*
**  - - - - - - - -
**   i a u P m p x
**  - - - - - - - -
**
**  Proper motion and parallax.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     rc,dc  double     ICRS RA,Dec at catalog epoch (radians)
**     pr     double     RA proper motion (radians/year, Note 1)
**     pd     double     Dec proper motion (radians/year)
**     px     double     parallax (arcsec)
**     rv     double     radial velocity (km/s, +ve if receding)
**     pmt    double     proper motion time interval (SSB, Julian years)
**     pob    double[3]  SSB to observer vector (au)
**
**  Returned:
**     pco    double[3]  coordinate direction (BCRS unit vector)
**
**  Notes:
**
**  1) The proper motion in RA is dRA/dt rather than cos(Dec)*dRA/dt.
**
**  2) The proper motion time interval is for when the starlight
**     reaches the solar system barycenter.
**
**  3) To avoid the need for iteration, the Roemer effect (i.e. the
**     small annual modulation of the proper motion coming from the
**     changing light time) is applied approximately, using the
**     direction of the star at the catalog epoch.
**
**  References:
**
**     1984 Astronomical Almanac, pp B39-B41.
**
**     Urban, S. & Seidelmann, P. K. (eds), Explanatory Supplement to
**     the Astronomical Almanac, 3rd ed., University Science Books
**     (2013), Section 7.2.
**
**  Called:
**     iauPdp       scalar product of two p-vectors
**     iauPn        decompose p-vector into modulus and direction
**
*/
```

```
int iauPmsafe(double ra1, double dec1, double pmr1, double pmd1,
              double px1, double rv1,
              double ep1a, double ep1b, double ep2a, double ep2b,
              double *ra2, double *dec2, double *pmr2, double *pmd2,
              double *px2, double *rv2)
/*
**  - - - - - - - - - -
**   i a u P m s a f e
**  - - - - - - - - - -
**
**  Star proper motion:  update star catalog data for space motion, with
**  special handling to handle the zero parallax case.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     ra1    double     right ascension (radians), before
**     dec1   double     declination (radians), before
**     pmr1   double     RA proper motion (radians/year), before
**     pmd1   double     Dec proper motion (radians/year), before
**     px1    double     parallax (arcseconds), before
**     rv1    double     radial velocity (km/s, +ve = receding), before
**     ep1a   double     "before" epoch, part A (Note 1)
**     ep1b   double     "before" epoch, part B (Note 1)
**     ep2a   double     "after" epoch, part A (Note 1)
**     ep2b   double     "after" epoch, part B (Note 1)
**
**  Returned:
**     ra2    double     right ascension (radians), after
**     dec2   double     declination (radians), after
**     pmr2   double     RA proper motion (radians/year), after
**     pmd2   double     Dec proper motion (radians/year), after
**     px2    double     parallax (arcseconds), after
**     rv2    double     radial velocity (km/s, +ve = receding), after
**
**  Returned (function value):
**            int        status:
**                          -1 = system error (should not occur)
**                           0 = no warnings or errors
**                           1 = distance overridden (Note 6)
**                           2 = excessive velocity (Note 7)
**                           4 = solution didn't converge (Note 8)
**                        else = binary logical OR of the above warnings
**
**  Notes:
**
**  1) The starting and ending TDB dates ep1a+ep1b and ep2a+ep2b are
**     Julian Dates, apportioned in any convenient way between the two
**     parts (A and B).  For example, JD(TDB)=2450123.7 could be
**     expressed in any of these ways, among others:
**
**            epNa            epNb
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in cases
**     where the loss of several decimal digits of resolution is
**     acceptable.  The J2000 method is best matched to the way the
**     argument is handled internally and will deliver the optimum
**     resolution.  The MJD method and the date & time methods are both
**     good compromises between resolution and convenience.
**
**  2) In accordance with normal star-catalog conventions, the object's
**     right ascension and declination are freed from the effects of
**     secular aberration.  The frame, which is aligned to the catalog
```

```
**     equator and equinox, is Lorentzian and centered on the SSB.
**
**     The proper motions are the rate of change of the right ascension
**     and declination at the catalog epoch and are in radians per TDB
**     Julian year.
**
**     The parallax and radial velocity are in the same frame.
**
**  3) Care is needed with units.  The star coordinates are in radians
**     and the proper motions in radians per Julian year, but the
**     parallax is in arcseconds.
**
**  4) The RA proper motion is in terms of coordinate angle, not true
**     angle.  If the catalog uses arcseconds for both RA and Dec proper
**     motions, the RA proper motion will need to be divided by cos(Dec)
**     before use.
**
**  5) Straight-line motion at constant speed, in the inertial frame, is
**     assumed.
**
**  6) An extremely small (or zero or negative) parallax is overridden
**     to ensure that the object is at a finite but very large distance,
**     but not so large that the proper motion is equivalent to a large
**     but safe speed (about 0.1c using the chosen constant).  A warning
**     status of 1 is added to the status if this action has been taken.
**
**  7) If the space velocity is a significant fraction of c (see the
**     constant VMAX in the function iauStarpv), it is arbitrarily set
**     to zero.  When this action occurs, 2 is added to the status.
**
**  8) The relativistic adjustment carried out in the iauStarpv function
**     involves an iterative calculation.  If the process fails to
**     converge within a set number of iterations, 4 is added to the
**     status.
**
**  Called:
**     iauSeps      angle between two points
**     iauStarpm    update star catalog data for space motion
**
*/
```

```
void iauPn(double p[3], double *r, double u[3])
/*
**  - - - - - -
**   i a u P n
**  - - - - - -
**
**  Convert a p-vector into modulus and unit vector.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     p        double[3]      p-vector
**
**  Returned:
**     r        double         modulus
**     u        double[3]      unit vector
**
**  Notes:
**
**  1) If p is null, the result is null.  Otherwise the result is a unit
**     vector.
**
**  2) It is permissible to re-use the same array for any of the
**     arguments.
**
**  Called:
**     iauPm        modulus of p-vector
**     iauZp        zero p-vector
**     iauSxp       multiply p-vector by scalar
**
*/
```

```
void iauPn00(double date1, double date2, double dpsi, double deps,
             double *epsa,
             double rb[3][3], double rp[3][3], double rbp[3][3],
             double rn[3][3], double rbpn[3][3])
/*
**  - - - - - - - -
**   i a u P n 0 0
**  - - - - - - - -
**
**  Precession-nutation, IAU 2000 model:  a multi-purpose function,
**  supporting classical (equinox-based) use directly and CIO-based
**  use indirectly.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1,date2  double          TT as a 2-part Julian Date (Note 1)
**     dpsi,deps    double          nutation (Note 2)
**
**  Returned:
**     epsa         double          mean obliquity (Note 3)
**     rb           double[3][3]    frame bias matrix (Note 4)
**     rp           double[3][3]    precession matrix (Note 5)
**     rbp          double[3][3]    bias-precession matrix (Note 6)
**     rn           double[3][3]    nutation matrix (Note 7)
**     rbpn         double[3][3]    GCRS-to-true matrix (Note 8)
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The caller is responsible for providing the nutation components;
**     they are in longitude and obliquity, in radians and are with
**     respect to the equinox and ecliptic of date.  For high-accuracy
**     applications, free core nutation should be included as well as
**     any other relevant corrections to the position of the CIP.
**
**  3) The returned mean obliquity is consistent with the IAU 2000
**     precession-nutation models.
**
**  4) The matrix rb transforms vectors from GCRS to J2000.0 mean
**     equator and equinox by applying frame bias.
**
**  5) The matrix rp transforms vectors from J2000.0 mean equator and
**     equinox to mean equator and equinox of date by applying
**     precession.
**
**  6) The matrix rbp transforms vectors from GCRS to mean equator and
**     equinox of date by applying frame bias then precession.  It is
**     the product rp x rb.
**
```

```
**  7) The matrix rn transforms vectors from mean equator and equinox of
**     date to true equator and equinox of date by applying the nutation
**     (luni-solar + planetary).
**
**  8) The matrix rbpn transforms vectors from GCRS to true equator and
**     equinox of date.  It is the product rn x rbp, applying frame
**     bias, precession and nutation in that order.
**
**  9) It is permissible to re-use the same array in the returned
**     arguments.  The arrays are filled in the order given.
**
**  Called:
**     iauPr00      IAU 2000 precession adjustments
**     iauObl80     mean obliquity, IAU 1980
**     iauBp00      frame bias and precession matrices, IAU 2000
**     iauCr        copy r-matrix
**     iauNumat     form nutation matrix
**     iauRxr       product of two r-matrices
**
**  Reference:
**
**     Capitaine, N., Chapront, J., Lambert, S. and Wallace, P.,
**     "Expressions for the Celestial Intermediate Pole and Celestial
**     Ephemeris Origin consistent with the IAU 2000A precession-
**     nutation model", Astron.Astrophys. 400, 1145-1154 (2003)
**
**     n.b. The celestial ephemeris origin (CEO) was renamed "celestial
**          intermediate origin" (CIO) by IAU 2006 Resolution 2.
**
*/
```

```
void iauPn00a(double date1, double date2,
              double *dpsi, double *deps, double *epsa,
              double rb[3][3], double rp[3][3], double rbp[3][3],
              double rn[3][3], double rbpn[3][3])
/*
**  - - - - - - - - -
**   i a u P n 0 0 a
**  - - - - - - - - -
**
**  Precession-nutation, IAU 2000A model:  a multi-purpose function,
**  supporting classical (equinox-based) use directly and CIO-based
**  use indirectly.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1,date2  double          TT as a 2-part Julian Date (Note 1)
**
**  Returned:
**     dpsi,deps    double          nutation (Note 2)
**     epsa         double          mean obliquity (Note 3)
**     rb           double[3][3]    frame bias matrix (Note 4)
**     rp           double[3][3]    precession matrix (Note 5)
**     rbp          double[3][3]    bias-precession matrix (Note 6)
**     rn           double[3][3]    nutation matrix (Note 7)
**     rbpn         double[3][3]    GCRS-to-true matrix (Notes 8,9)
**
**  Notes:
**
**  1)  The TT date date1+date2 is a Julian Date, apportioned in any
**      convenient way between the two arguments.  For example,
**      JD(TT)=2450123.7 could be expressed in any of these ways,
**      among others:
**
**             date1          date2
**
**          2450123.7           0.0       (JD method)
**          2451545.0       -1421.3       (J2000 method)
**          2400000.5       50123.2       (MJD method)
**          2450123.5           0.2       (date & time method)
**
**      The JD method is the most natural and convenient to use in
**      cases where the loss of several decimal digits of resolution
**      is acceptable.  The J2000 method is best matched to the way
**      the argument is handled internally and will deliver the
**      optimum resolution.  The MJD method and the date & time methods
**      are both good compromises between resolution and convenience.
**
**  2)  The nutation components (luni-solar + planetary, IAU 2000A) in
**      longitude and obliquity are in radians and with respect to the
**      equinox and ecliptic of date.  Free core nutation is omitted;
**      for the utmost accuracy, use the iauPn00 function, where the
**      nutation components are caller-specified.  For faster but
**      slightly less accurate results, use the iauPn00b function.
**
**  3)  The mean obliquity is consistent with the IAU 2000 precession.
**
**  4)  The matrix rb transforms vectors from GCRS to J2000.0 mean
**      equator and equinox by applying frame bias.
**
**  5)  The matrix rp transforms vectors from J2000.0 mean equator and
**      equinox to mean equator and equinox of date by applying
**      precession.
**
**  6)  The matrix rbp transforms vectors from GCRS to mean equator and
**      equinox of date by applying frame bias then precession.  It is
**      the product rp x rb.
**
```

```
**  7)  The matrix rn transforms vectors from mean equator and equinox
**      of date to true equator and equinox of date by applying the
**      nutation (luni-solar + planetary).
**
**  8)  The matrix rbpn transforms vectors from GCRS to true equator and
**      equinox of date.  It is the product rn x rbp, applying frame
**      bias, precession and nutation in that order.
**
**  9)  The X,Y,Z coordinates of the IAU 2000A Celestial Intermediate
**      Pole are elements (3,1-3) of the GCRS-to-true matrix,
**      i.e. rbpn[2][0-2].
**
**  10) It is permissible to re-use the same array in the returned
**      arguments.  The arrays are filled in the stated order.
**
**  Called:
**     iauNut00a    nutation, IAU 2000A
**     iauPn00      bias/precession/nutation results, IAU 2000
**
**  Reference:
**
**     Capitaine, N., Chapront, J., Lambert, S. and Wallace, P.,
**     "Expressions for the Celestial Intermediate Pole and Celestial
**     Ephemeris Origin consistent with the IAU 2000A precession-
**     nutation model", Astron.Astrophys. 400, 1145-1154 (2003)
**
**     n.b. The celestial ephemeris origin (CEO) was renamed "celestial
**          intermediate origin" (CIO) by IAU 2006 Resolution 2.
**
*/
```

```
void iauPn00b(double date1, double date2,
              double *dpsi, double *deps, double *epsa,
              double rb[3][3], double rp[3][3], double rbp[3][3],
              double rn[3][3], double rbpn[3][3])
/*
**  - - - - - - - - -
**   i a u P n 0 0 b
**  - - - - - - - - -
**
**  Precession-nutation, IAU 2000B model:  a multi-purpose function,
**  supporting classical (equinox-based) use directly and CIO-based
**  use indirectly.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1,date2  double          TT as a 2-part Julian Date (Note 1)
**
**  Returned:
**     dpsi,deps    double          nutation (Note 2)
**     epsa         double          mean obliquity (Note 3)
**     rb           double[3][3]    frame bias matrix (Note 4)
**     rp           double[3][3]    precession matrix (Note 5)
**     rbp          double[3][3]    bias-precession matrix (Note 6)
**     rn           double[3][3]    nutation matrix (Note 7)
**     rbpn         double[3][3]    GCRS-to-true matrix (Notes 8,9)
**
**  Notes:
**
**  1)  The TT date date1+date2 is a Julian Date, apportioned in any
**      convenient way between the two arguments.  For example,
**      JD(TT)=2450123.7 could be expressed in any of these ways,
**      among others:
**
**             date1          date2
**
**          2450123.7           0.0       (JD method)
**          2451545.0       -1421.3       (J2000 method)
**          2400000.5       50123.2       (MJD method)
**          2450123.5           0.2       (date & time method)
**
**      The JD method is the most natural and convenient to use in
**      cases where the loss of several decimal digits of resolution
**      is acceptable.  The J2000 method is best matched to the way
**      the argument is handled internally and will deliver the
**      optimum resolution.  The MJD method and the date & time methods
**      are both good compromises between resolution and convenience.
**
**  2)  The nutation components (luni-solar + planetary, IAU 2000B) in
**      longitude and obliquity are in radians and with respect to the
**      equinox and ecliptic of date.  For more accurate results, but
**      at the cost of increased computation, use the iauPn00a function.
**      For the utmost accuracy, use the iauPn00 function, where the
**      nutation components are caller-specified.
**
**  3)  The mean obliquity is consistent with the IAU 2000 precession.
**
**  4)  The matrix rb transforms vectors from GCRS to J2000.0 mean
**      equator and equinox by applying frame bias.
**
**  5)  The matrix rp transforms vectors from J2000.0 mean equator and
**      equinox to mean equator and equinox of date by applying
**      precession.
**
**  6)  The matrix rbp transforms vectors from GCRS to mean equator and
**      equinox of date by applying frame bias then precession.  It is
**      the product rp x rb.
**
```

```
**  7)  The matrix rn transforms vectors from mean equator and equinox
**      of date to true equator and equinox of date by applying the
**      nutation (luni-solar + planetary).
**
**  8)  The matrix rbpn transforms vectors from GCRS to true equator and
**      equinox of date.  It is the product rn x rbp, applying frame
**      bias, precession and nutation in that order.
**
**  9)  The X,Y,Z coordinates of the IAU 2000B Celestial Intermediate
**      Pole are elements (3,1-3) of the GCRS-to-true matrix,
**      i.e. rbpn[2][0-2].
**
**  10) It is permissible to re-use the same array in the returned
**      arguments.  The arrays are filled in the stated order.
**
**  Called:
**     iauNut00b    nutation, IAU 2000B
**     iauPn00      bias/precession/nutation results, IAU 2000
**
**  Reference:
**
**     Capitaine, N., Chapront, J., Lambert, S. and Wallace, P.,
**     "Expressions for the Celestial Intermediate Pole and Celestial
**     Ephemeris Origin consistent with the IAU 2000A precession-
**     nutation model", Astron.Astrophys. 400, 1145-1154 (2003).
**
**     n.b. The celestial ephemeris origin (CEO) was renamed "celestial
**          intermediate origin" (CIO) by IAU 2006 Resolution 2.
**
*/
```

```
void iauPn06(double date1, double date2, double dpsi, double deps,
             double *epsa,
             double rb[3][3], double rp[3][3], double rbp[3][3],
             double rn[3][3], double rbpn[3][3])
/*
**  - - - - - - - -
**   i a u P n 0 6
**  - - - - - - - -
**
**  Precession-nutation, IAU 2006 model:  a multi-purpose function,
**  supporting classical (equinox-based) use directly and CIO-based use
**  indirectly.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1,date2  double          TT as a 2-part Julian Date (Note 1)
**     dpsi,deps    double          nutation (Note 2)
**
**  Returned:
**     epsa         double          mean obliquity (Note 3)
**     rb           double[3][3]    frame bias matrix (Note 4)
**     rp           double[3][3]    precession matrix (Note 5)
**     rbp          double[3][3]    bias-precession matrix (Note 6)
**     rn           double[3][3]    nutation matrix (Note 7)
**     rbpn         double[3][3]    GCRS-to-true matrix (Notes 8,9)
**
**  Notes:
**
**  1)  The TT date date1+date2 is a Julian Date, apportioned in any
**      convenient way between the two arguments.  For example,
**      JD(TT)=2450123.7 could be expressed in any of these ways,
**      among others:
**
**             date1          date2
**
**          2450123.7           0.0       (JD method)
**          2451545.0       -1421.3       (J2000 method)
**          2400000.5       50123.2       (MJD method)
**          2450123.5           0.2       (date & time method)
**
**      The JD method is the most natural and convenient to use in
**      cases where the loss of several decimal digits of resolution
**      is acceptable.  The J2000 method is best matched to the way
**      the argument is handled internally and will deliver the
**      optimum resolution.  The MJD method and the date & time methods
**      are both good compromises between resolution and convenience.
**
**  2)  The caller is responsible for providing the nutation components;
**      they are in longitude and obliquity, in radians and are with
**      respect to the equinox and ecliptic of date.  For high-accuracy
**      applications, free core nutation should be included as well as
**      any other relevant corrections to the position of the CIP.
**
**  3)  The returned mean obliquity is consistent with the IAU 2006
**      precession.
**
**  4)  The matrix rb transforms vectors from GCRS to J2000.0 mean
**      equator and equinox by applying frame bias.
**
**  5)  The matrix rp transforms vectors from J2000.0 mean equator and
**      equinox to mean equator and equinox of date by applying
**      precession.
**
**  6)  The matrix rbp transforms vectors from GCRS to mean equator and
**      equinox of date by applying frame bias then precession.  It is
**      the product rp x rb.
**
```

```
**  7)  The matrix rn transforms vectors from mean equator and equinox
**      of date to true equator and equinox of date by applying the
**      nutation (luni-solar + planetary).
**
**  8)  The matrix rbpn transforms vectors from GCRS to true equator and
**      equinox of date.  It is the product rn x rbp, applying frame
**      bias, precession and nutation in that order.
**
**  9)  The X,Y,Z coordinates of the Celestial Intermediate Pole are
**      elements (3,1-3) of the GCRS-to-true matrix, i.e. rbpn[2][0-2].
**
**  10) It is permissible to re-use the same array in the returned
**      arguments.  The arrays are filled in the stated order.
**
**  Called:
**     iauPfw06     bias-precession F-W angles, IAU 2006
**     iauFw2m      F-W angles to r-matrix
**     iauCr        copy r-matrix
**     iauTr        transpose r-matrix
**     iauRxr       product of two r-matrices
**
**  References:
**
**     Capitaine, N. & Wallace, P.T., 2006, Astron.Astrophys. 450, 855
**
**     Wallace, P.T. & Capitaine, N., 2006, Astron.Astrophys. 459, 981
**
*/
```

```
void iauPn06a(double date1, double date2,
              double *dpsi, double *deps, double *epsa,
              double rb[3][3], double rp[3][3], double rbp[3][3],
              double rn[3][3], double rbpn[3][3])
/*
**  - - - - - - - - -
**   i a u P n 0 6 a
**  - - - - - - - - -
**
**  Precession-nutation, IAU 2006/2000A models:  a multi-purpose function,
**  supporting classical (equinox-based) use directly and CIO-based use
**  indirectly.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1,date2  double          TT as a 2-part Julian Date (Note 1)
**
**  Returned:
**     dpsi,deps    double          nutation (Note 2)
**     epsa         double          mean obliquity (Note 3)
**     rb           double[3][3]    frame bias matrix (Note 4)
**     rp           double[3][3]    precession matrix (Note 5)
**     rbp          double[3][3]    bias-precession matrix (Note 6)
**     rn           double[3][3]    nutation matrix (Note 7)
**     rbpn         double[3][3]    GCRS-to-true matrix (Notes 8,9)
**
**  Notes:
**
**  1)  The TT date date1+date2 is a Julian Date, apportioned in any
**      convenient way between the two arguments.  For example,
**      JD(TT)=2450123.7 could be expressed in any of these ways,
**      among others:
**
**             date1          date2
**
**          2450123.7           0.0       (JD method)
**          2451545.0       -1421.3       (J2000 method)
**          2400000.5       50123.2       (MJD method)
**          2450123.5           0.2       (date & time method)
**
**      The JD method is the most natural and convenient to use in
**      cases where the loss of several decimal digits of resolution
**      is acceptable.  The J2000 method is best matched to the way
**      the argument is handled internally and will deliver the
**      optimum resolution.  The MJD method and the date & time methods
**      are both good compromises between resolution and convenience.
**
**  2)  The nutation components (luni-solar + planetary, IAU 2000A) in
**      longitude and obliquity are in radians and with respect to the
**      equinox and ecliptic of date.  Free core nutation is omitted;
**      for the utmost accuracy, use the iauPn06 function, where the
**      nutation components are caller-specified.
**
**  3)  The mean obliquity is consistent with the IAU 2006 precession.
**
**  4)  The matrix rb transforms vectors from GCRS to mean J2000.0 by
**      applying frame bias.
**
**  5)  The matrix rp transforms vectors from mean J2000.0 to mean of
**      date by applying precession.
**
**  6)  The matrix rbp transforms vectors from GCRS to mean of date by
**      applying frame bias then precession.  It is the product rp x rb.
**
**  7)  The matrix rn transforms vectors from mean of date to true of
**      date by applying the nutation (luni-solar + planetary).
**
```

```
**  8)  The matrix rbpn transforms vectors from GCRS to true of date
**      (CIP/equinox).  It is the product rn x rbp, applying frame bias,
**      precession and nutation in that order.
**
**  9)  The X,Y,Z coordinates of the IAU 2006/2000A Celestial
**      Intermediate Pole are elements (3,1-3) of the GCRS-to-true
**      matrix, i.e. rbpn[2][0-2].
**
**  10) It is permissible to re-use the same array in the returned
**      arguments.  The arrays are filled in the stated order.
**
**  Called:
**     iauNut06a    nutation, IAU 2006/2000A
**     iauPn06      bias/precession/nutation results, IAU 2006
**
**  Reference:
**
**     Capitaine, N. & Wallace, P.T., 2006, Astron.Astrophys. 450, 855
**
*/
```

```
void iauPnm00a(double date1, double date2, double rbpn[3][3])
/*
**  - - - - - - - - - -
**   i a u P n m 0 0 a
**  - - - - - - - - - -
**
**  Form the matrix of precession-nutation for a given date (including
**  frame bias), equinox based, IAU 2000A model.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1,date2 double       TT as a 2-part Julian Date (Note 1)
**
**  Returned:
**     rbpn        double[3][3] bias-precession-nutation matrix (Note 2)
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways, among
**     others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The matrix operates in the sense V(date) = rbpn * V(GCRS), where
**     the p-vector V(date) is with respect to the true equatorial triad
**     of date date1+date2 and the p-vector V(GCRS) is with respect to
**     the Geocentric Celestial Reference System (IAU, 2000).
**
**  3) A faster, but slightly less accurate, result (about 1 mas) can be
**     obtained by using instead the iauPnm00b function.
**
**  Called:
**     iauPn00a     bias/precession/nutation, IAU 2000A
**
**  Reference:
**
**     IAU: Trans. International Astronomical Union, Vol. XXIVB;  Proc.
**     24th General Assembly, Manchester, UK.  Resolutions B1.3, B1.6.
**     (2000)
**
*/
```

```
void iauPnm00b(double date1, double date2, double rbpn[3][3])
/*
**  - - - - - - - - - -
**   i a u P n m 0 0 b
**  - - - - - - - - - -
**
**  Form the matrix of precession-nutation for a given date (including
**  frame bias), equinox-based, IAU 2000B model.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1,date2 double       TT as a 2-part Julian Date (Note 1)
**
**  Returned:
**     rbpn        double[3][3] bias-precession-nutation matrix (Note 2)
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways, among
**     others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The matrix operates in the sense V(date) = rbpn * V(GCRS), where
**     the p-vector V(date) is with respect to the true equatorial triad
**     of date date1+date2 and the p-vector V(GCRS) is with respect to
**     the Geocentric Celestial Reference System (IAU, 2000).
**
**  3) The present function is faster, but slightly less accurate (about
**     1 mas), than the iauPnm00a function.
**
**  Called:
**     iauPn00b     bias/precession/nutation, IAU 2000B
**
**  Reference:
**
**     IAU: Trans. International Astronomical Union, Vol. XXIVB;  Proc.
**     24th General Assembly, Manchester, UK.  Resolutions B1.3, B1.6.
**     (2000)
**
*/
```

```
void iauPnm06a(double date1, double date2, double rbpn[3][3])
/*
**  - - - - - - - - - -
**   i a u P n m 0 6 a
**  - - - - - - - - - -
**
**  Form the matrix of precession-nutation for a given date (including
**  frame bias), equinox based, IAU 2006 precession and IAU 2000A
**  nutation models.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1,date2 double       TT as a 2-part Julian Date (Note 1)
**
**  Returned:
**     rbpn        double[3][3] bias-precession-nutation matrix (Note 2)
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways, among
**     others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The matrix operates in the sense V(date) = rbpn * V(GCRS), where
**     the p-vector V(date) is with respect to the true equatorial triad
**     of date date1+date2 and the p-vector V(GCRS) is with respect to
**     the Geocentric Celestial Reference System (IAU, 2000).
**
**  Called:
**     iauPfw06     bias-precession F-W angles, IAU 2006
**     iauNut06a    nutation, IAU 2006/2000A
**     iauFw2m      F-W angles to r-matrix
**
**  Reference:
**
**     Capitaine, N. & Wallace, P.T., 2006, Astron.Astrophys. 450, 855.
**
*/
```

```
void iauPnm80(double date1, double date2, double rmatpn[3][3])
/*
**  - - - - - - - - - -
**   i a u P n m 8 0
**  - - - - - - - - - -
**
**  Form the matrix of precession/nutation for a given date, IAU 1976
**  precession model, IAU 1980 nutation model.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1,date2 double       TT as a 2-part Julian Date (Note 1)
**
**  Returned:
**     rmatpn         double[3][3]   combined precession/nutation matrix
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The matrix operates in the sense V(date) = rmatpn * V(J2000),
**     where the p-vector V(date) is with respect to the true equatorial
**     triad of date date1+date2 and the p-vector V(J2000) is with
**     respect to the mean equatorial triad of epoch J2000.0.
**
**  Called:
**     iauPmat76    precession matrix, IAU 1976
**     iauNutm80    nutation matrix, IAU 1980
**     iauRxr       product of two r-matrices
**
**  Reference:
**
**     Explanatory Supplement to the Astronomical Almanac,
**     P. Kenneth Seidelmann (ed), University Science Books (1992),
**     Section 3.3 (p145).
**
*/
```

```
void iauPom00(double xp, double yp, double sp, double rpom[3][3])
/*
**  - - - - - - - - - -
**   i a u P o m 0 0
**  - - - - - - - - - -
**
**  Form the matrix of polar motion for a given date, IAU 2000.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     xp,yp    double    coordinates of the pole (radians, Note 1)
**     sp       double    the TIO locator s' (radians, Note 2)
**
**  Returned:
**     rpom     double[3][3]   polar-motion matrix (Note 3)
**
**  Notes:
**
**  1) The arguments xp and yp are the coordinates (in radians) of the
**     Celestial Intermediate Pole with respect to the International
**     Terrestrial Reference System (see IERS Conventions 2003),
**     measured along the meridians 0 and 90 deg west respectively.
**
**  2) The argument sp is the TIO locator s', in radians, which
**     positions the Terrestrial Intermediate Origin on the equator.  It
**     is obtained from polar motion observations by numerical
**     integration, and so is in essence unpredictable.  However, it is
**     dominated by a secular drift of about 47 microarcseconds per
**     century, and so can be taken into account by using s' = -47*t,
**     where t is centuries since J2000.0.  The function iauSp00
**     implements this approximation.
**
**  3) The matrix operates in the sense V(TRS) = rpom * V(CIP), meaning
**     that it is the final rotation when computing the pointing
**     direction to a celestial source.
**
**  Called:
**     iauIr        initialize r-matrix to identity
**     iauRz        rotate around Z-axis
**     iauRy        rotate around Y-axis
**     iauRx        rotate around X-axis
**
**  Reference:
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG (2004)
**
*/
```

```
void iauPpp(double a[3], double b[3], double apb[3])
/*
**  - - - - - - -
**   i a u P p p
**  - - - - - - -
**
**  P-vector addition.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     a        double[3]      first p-vector
**     b        double[3]      second p-vector
**
**  Returned:
**     apb      double[3]      a + b
**
**  Note:
**     It is permissible to re-use the same array for any of the
**     arguments.
**
*/
```

```
void iauPpsp(double a[3], double s, double b[3], double apsb[3])
/*
**  - - - - - - - - -
**   i a u P p s p
**  - - - - - - - - -
**
**  P-vector plus scaled p-vector.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     a      double[3]     first p-vector
**     s      double        scalar (multiplier for b)
**     b      double[3]     second p-vector
**
**  Returned:
**     apsb   double[3]     a + s*b
**
**  Note:
**     It is permissible for any of a, b and apsb to be the same array.
**
**  Called:
**     iauSxp       multiply p-vector by scalar
**     iauPpp       p-vector plus p-vector
**
*/
```

```
void iauPr00(double date1, double date2, double *dpsipr, double *depspr)
/*
**  - - - - - - - -
**   i a u P r 0 0
**  - - - - - - - -
**
**  Precession-rate part of the IAU 2000 precession-nutation models
**  (part of MHB2000).
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical model.
**
**  Given:
**     date1,date2    double  TT as a 2-part Julian Date (Note 1)
**
**  Returned:
**     dpsipr,depspr  double  precession corrections (Notes 2,3)
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The precession adjustments are expressed as "nutation
**     components", corrections in longitude and obliquity with respect
**     to the J2000.0 equinox and ecliptic.
**
**  3) Although the precession adjustments are stated to be with respect
**     to Lieske et al. (1977), the MHB2000 model does not specify which
**     set of Euler angles are to be used and how the adjustments are to
**     be applied.  The most literal and straightforward procedure is to
**     adopt the 4-rotation epsilon_0, psi_A, omega_A, xi_A option, and
**     to add dpsipr to psi_A and depspr to both omega_A and eps_A.
**
**  4) This is an implementation of one aspect of the IAU 2000A nutation
**     model, formally adopted by the IAU General Assembly in 2000,
**     namely MHB2000 (Mathews et al. 2002).
**
**  References:
**
**     Lieske, J.H., Lederle, T., Fricke, W. & Morando, B., "Expressions
**     for the precession quantities based upon the IAU (1976) System of
**     Astronomical Constants", Astron.Astrophys., 58, 1-16 (1977)
**
**     Mathews, P.M., Herring, T.A., Buffet, B.A., "Modeling of nutation
**     and precession   New nutation series for nonrigid Earth and
**     insights into the Earth's interior", J.Geophys.Res., 107, B4,
**     2002.  The MHB2000 code itself was obtained on 9th September 2002
**     from ftp://maia.usno.navy.mil/conv2000/chapter5/IAU2000A.
**
**     Wallace, P.T., "Software for Implementing the IAU 2000
**     Resolutions", in IERS Workshop 5.1 (2002).
**
```

```
*/
```

```
void iauPrec76(double date01, double date02, double date11, double date12,
               double *zeta, double *z, double *theta)
/*
**  - - - - - - - - - -
**   i a u P r e c 7 6
**  - - - - - - - - - -
**
**  IAU 1976 precession model.
**
**  This function forms the three Euler angles which implement general
**  precession between two dates, using the IAU 1976 model (as for the
**  FK5 catalog).
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical model.
**
**  Given:
**     date01,date02   double    TDB starting date (Note 1)
**     date11,date12   double    TDB ending date (Note 1)
**
**  Returned:
**     zeta            double    1st rotation: radians cw around z
**     z               double    3rd rotation: radians cw around z
**     theta           double    2nd rotation: radians ccw around y
**
**  Notes:
**
**  1) The dates date01+date02 and date11+date12 are Julian Dates,
**     apportioned in any convenient way between the arguments daten1
**     and daten2.  For example, JD(TDB)=2450123.7 could be expressed in
**     any of these ways, among others:
**
**           daten1        daten2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in cases
**     where the loss of several decimal digits of resolution is
**     acceptable.  The J2000 method is best matched to the way the
**     argument is handled internally and will deliver the optimum
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**     The two dates may be expressed using different methods, but at
**     the risk of losing some resolution.
**
**  2) The accumulated precession angles zeta, z, theta are expressed
**     through canonical polynomials which are valid only for a limited
**     time span.  In addition, the IAU 1976 precession rate is known to
**     be imperfect.  The absolute accuracy of the present formulation
**     is better than 0.1 arcsec from 1960AD to 2040AD, better than
**     1 arcsec from 1640AD to 2360AD, and remains below 3 arcsec for
**     the whole of the period 500BC to 3000AD.  The errors exceed
**     10 arcsec outside the range 1200BC to 3900AD, exceed 100 arcsec
**     outside 4200BC to 5600AD and exceed 1000 arcsec outside 6800BC to
**     8200AD.
**
**  3) The three angles are returned in the conventional order, which
**     is not the same as the order of the corresponding Euler
**     rotations.  The precession matrix is
**     R_3(-z) x R_2(+theta) x R_3(-zeta).
**
**  Reference:
**
**     Lieske, J.H., 1979, Astron.Astrophys. 73, 282, equations
**     (6) & (7), p283.
**
```

```
*/
```

```
void iauPv2p(double pv[2][3], double p[3])
/*
**  - - - - - - - -
**   i a u P v 2 p
**  - - - - - - - -
**
**  Discard velocity component of a pv-vector.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     pv      double[2][3]     pv-vector
**
**  Returned:
**     p       double[3]        p-vector
**
**  Called:
**     iauCp        copy p-vector
**
*/
```

```
void iauPv2s(double pv[2][3],
             double *theta, double *phi, double *r,
             double *td, double *pd, double *rd)
/*
**  - - - - - - - -
**   i a u P v 2 s
**  - - - - - - - -
**
**  Convert position/velocity from Cartesian to spherical coordinates.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     pv       double[2][3]  pv-vector
**
**  Returned:
**     theta    double        longitude angle (radians)
**     phi      double        latitude angle (radians)
**     r        double        radial distance
**     td       double        rate of change of theta
**     pd       double        rate of change of phi
**     rd       double        rate of change of r
**
**  Notes:
**
**  1) If the position part of pv is null, theta, phi, td and pd
**     are indeterminate.  This is handled by extrapolating the
**     position through unit time by using the velocity part of
**     pv.  This moves the origin without changing the direction
**     of the velocity component.  If the position and velocity
**     components of pv are both null, zeroes are returned for all
**     six results.
**
**  2) If the position is a pole, theta, td and pd are indeterminate.
**     In such cases zeroes are returned for all three.
**
*/
```

```
void iauPvdpv(double a[2][3], double b[2][3], double adb[2])
/*
**  - - - - - - - - -
**   i a u P v d p v
**  - - - - - - - - -
**
**  Inner (=scalar=dot) product of two pv-vectors.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     a        double[2][3]      first pv-vector
**     b        double[2][3]      second pv-vector
**
**  Returned:
**     adb      double[2]         a . b (see note)
**
**  Note:
**
**     If the position and velocity components of the two pv-vectors are
**     ( ap, av ) and ( bp, bv ), the result, a . b, is the pair of
**     numbers ( ap . bp , ap . bv + av . bp ).  The two numbers are the
**     dot-product of the two p-vectors and its derivative.
**
**  Called:
**     iauPdp       scalar product of two p-vectors
**
*/
```

```
void iauPvm(double pv[2][3], double *r, double *s)
/*
**  - - - - - - -
**   i a u P v m
**  - - - - - - -
**
**  Modulus of pv-vector.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     pv     double[2][3]   pv-vector
**
**  Returned:
**     r      double         modulus of position component
**     s      double         modulus of velocity component
**
**  Called:
**     iauPm        modulus of p-vector
**
*/
```

```
void iauPvmpv(double a[2][3], double b[2][3], double amb[2][3])
/*
**  - - - - - - - - -
**   i a u P v m p v
**  - - - - - - - - -
**
**  Subtract one pv-vector from another.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     a       double[2][3]      first pv-vector
**     b       double[2][3]      second pv-vector
**
**  Returned:
**     amb     double[2][3]      a - b
**
**  Note:
**     It is permissible to re-use the same array for any of the
**     arguments.
**
**  Called:
**     iauPmp       p-vector minus p-vector
**
*/
```

```
void iauPvppv(double a[2][3], double b[2][3], double apb[2][3])
/*
**  - - - - - - - - -
**   i a u P v p p v
**  - - - - - - - - -
**
**  Add one pv-vector to another.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     a        double[2][3]      first pv-vector
**     b        double[2][3]      second pv-vector
**
**  Returned:
**     apb      double[2][3]      a + b
**
**  Note:
**     It is permissible to re-use the same array for any of the
**     arguments.
**
**  Called:
**     iauPpp       p-vector plus p-vector
**
*/
```

```
int iauPvstar(double pv[2][3], double *ra, double *dec,
              double *pmr, double *pmd, double *px, double *rv)
/*
**  - - - - - - - - - -
**   i a u P v s t a r
**  - - - - - - - - - -
**
**  Convert star position+velocity vector to catalog coordinates.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given (Note 1):
**     pv     double[2][3]   pv-vector (au, au/day)
**
**  Returned (Note 2):
**     ra     double         right ascension (radians)
**     dec    double         declination (radians)
**     pmr    double         RA proper motion (radians/year)
**     pmd    double         Dec proper motion (radians/year)
**     px     double         parallax (arcsec)
**     rv     double         radial velocity (km/s, positive = receding)
**
**  Returned (function value):
**            int            status:
**                              0 = OK
**                             -1 = superluminal speed (Note 5)
**                             -2 = null position vector
**
**  Notes:
**
**  1) The specified pv-vector is the coordinate direction (and its rate
**     of change) for the date at which the light leaving the star
**     reached the solar-system barycenter.
**
**  2) The star data returned by this function are "observables" for an
**     imaginary observer at the solar-system barycenter.  Proper motion
**     and radial velocity are, strictly, in terms of barycentric
**     coordinate time, TCB.  For most practical applications, it is
**     permissible to neglect the distinction between TCB and ordinary
**     "proper" time on Earth (TT/TAI).  The result will, as a rule, be
**     limited by the intrinsic accuracy of the proper-motion and
**     radial-velocity data;  moreover, the supplied pv-vector is likely
**     to be merely an intermediate result (for example generated by the
**     function iauStarpv), so that a change of time unit will cancel
**     out overall.
**
**     In accordance with normal star-catalog conventions, the object's
**     right ascension and declination are freed from the effects of
**     secular aberration.  The frame, which is aligned to the catalog
**     equator and equinox, is Lorentzian and centered on the SSB.
**
**     Summarizing, the specified pv-vector is for most stars almost
**     identical to the result of applying the standard geometrical
**     "space motion" transformation to the catalog data.  The
**     differences, which are the subject of the Stumpff paper cited
**     below, are:
**
**     (i) In stars with significant radial velocity and proper motion,
**     the constantly changing light-time distorts the apparent proper
**     motion.  Note that this is a classical, not a relativistic,
**     effect.
**
**     (ii) The transformation complies with special relativity.
**
**  3) Care is needed with units.  The star coordinates are in radians
**     and the proper motions in radians per Julian year, but the
**     parallax is in arcseconds; the radial velocity is in km/s, but
**     the pv-vector result is in au and au/day.
```

```
**
**  4) The proper motions are the rate of change of the right ascension
**     and declination at the catalog epoch and are in radians per Julian
**     year.  The RA proper motion is in terms of coordinate angle, not
**     true angle, and will thus be numerically larger at high
**     declinations.
**
**  5) Straight-line motion at constant speed in the inertial frame is
**     assumed.  If the speed is greater than or equal to the speed of
**     light, the function aborts with an error status.
**
**  6) The inverse transformation is performed by the function iauStarpv.
**
**  Called:
**     iauPn        decompose p-vector into modulus and direction
**     iauPdp       scalar product of two p-vectors
**     iauSxp       multiply p-vector by scalar
**     iauPmp       p-vector minus p-vector
**     iauPm        modulus of p-vector
**     iauPpp       p-vector plus p-vector
**     iauPv2s      pv-vector to spherical
**     iauAnp       normalize angle into range 0 to 2pi
**
**  Reference:
**
**     Stumpff, P., 1985, Astron.Astrophys. 144, 232-240.
**
*/
```

```
void iauPvtob(double elong, double phi, double hm,
              double xp, double yp, double sp, double theta,
              double pv[2][3])
/*
**  - - - - - - - - -
**   i a u P v t o b
**  - - - - - - - - -
**
**  Position and velocity of a terrestrial observing station.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     elong   double       longitude (radians, east +ve, Note 1)
**     phi     double       latitude (geodetic, radians, Note 1)
**     hm      double       height above ref. ellipsoid (geodetic, m)
**     xp,yp   double       coordinates of the pole (radians, Note 2)
**     sp      double       the TIO locator s' (radians, Note 2)
**     theta   double       Earth rotation angle (radians, Note 3)
**
**  Returned:
**     pv      double[2][3] position/velocity vector (m, m/s, CIRS)
**
**  Notes:
**
**  1) The terrestrial coordinates are with respect to the WGS84
**     reference ellipsoid.
**
**  2) xp and yp are the coordinates (in radians) of the Celestial
**     Intermediate Pole with respect to the International Terrestrial
**     Reference System (see IERS Conventions), measured along the
**     meridians 0 and 90 deg west respectively.  sp is the TIO locator
**     s', in radians, which positions the Terrestrial Intermediate
**     Origin on the equator.  For many applications, xp, yp and
**     (especially) sp can be set to zero.
**
**  3) If theta is Greenwich apparent sidereal time instead of Earth
**     rotation angle, the result is with respect to the true equator
**     and equinox of date, i.e. with the x-axis at the equinox rather
**     than the celestial intermediate origin.
**
**  4) The velocity units are meters per UT1 second, not per SI second.
**     This is unlikely to have any practical consequences in the modern
**     era.
**
**  5) No validation is performed on the arguments.  Error cases that
**     could lead to arithmetic exceptions are trapped by the iauGd2gc
**     function, and the result set to zeros.
**
**  References:
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG (2004)
**
**     Urban, S. & Seidelmann, P. K. (eds), Explanatory Supplement to
**     the Astronomical Almanac, 3rd ed., University Science Books
**     (2013), Section 7.4.3.3.
**
**  Called:
**     iauGd2gc     geodetic to geocentric transformation
**     iauPom00     polar motion matrix
**     iauTrxp      product of transpose of r-matrix and p-vector
**
*/
```

```
void iauPvu(double dt, double pv[2][3], double upv[2][3])
/*
**  - - - - - - -
**   i a u P v u
**  - - - - - - -
**
**  Update a pv-vector.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     dt       double           time interval
**     pv       double[2][3]     pv-vector
**
**  Returned:
**     upv      double[2][3]     p updated, v unchanged
**
**  Notes:
**
**  1) "Update" means "refer the position component of the vector
**     to a new date dt time units from the existing date".
**
**  2) The time units of dt must match those of the velocity.
**
**  3) It is permissible for pv and upv to be the same array.
**
**  Called:
**     iauPpsp      p-vector plus scaled p-vector
**     iauCp        copy p-vector
**
*/
```

```
void iauPvup(double dt, double pv[2][3], double p[3])
/*
**  - - - - - - - -
**   i a u P v u p
**  - - - - - - - -
**
**  Update a pv-vector, discarding the velocity component.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     dt       double            time interval
**     pv       double[2][3]      pv-vector
**
**  Returned:
**     p        double[3]         p-vector
**
**  Notes:
**
**  1) "Update" means "refer the position component of the vector to a
**     new date dt time units from the existing date".
**
**  2) The time units of dt must match those of the velocity.
**
*/
```

```
void iauPvxpv(double a[2][3], double b[2][3], double axb[2][3])
/*
**  - - - - - - - - -
**   i a u P v x p v
**  - - - - - - - - -
**
**  Outer (=vector=cross) product of two pv-vectors.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     a        double[2][3]      first pv-vector
**     b        double[2][3]      second pv-vector
**
**  Returned:
**     axb      double[2][3]      a x b
**
**  Notes:
**
**  1) If the position and velocity components of the two pv-vectors are
**     ( ap, av ) and ( bp, bv ), the result, a x b, is the pair of
**     vectors ( ap x bp, ap x bv + av x bp ).  The two vectors are the
**     cross-product of the two p-vectors and its derivative.
**
**  2) It is permissible to re-use the same array for any of the
**     arguments.
**
**  Called:
**     iauCpv       copy pv-vector
**     iauPxp       vector product of two p-vectors
**     iauPpp       p-vector plus p-vector
**
*/
```

```
void iauPxp(double a[3], double b[3], double axb[3])
/*
**  - - - - - - -
**   i a u P x p
**  - - - - - - -
**
**  p-vector outer (=vector=cross) product.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     a        double[3]      first p-vector
**     b        double[3]      second p-vector
**
**  Returned:
**     axb      double[3]      a x b
**
**  Note:
**     It is permissible to re-use the same array for any of the
**     arguments.
**
*/
```

```
void iauRefco(double phpa, double tc, double rh, double wl,
              double *refa, double *refb)
/*
**  - - - - - - - - -
**   i a u R e f c o
**  - - - - - - - - -
**
**  Determine the constants A and B in the atmospheric refraction model
**  dZ = A tan Z + B tan^3 Z.
**
**  Z is the "observed" zenith distance (i.e. affected by refraction)
**  and dZ is what to add to Z to give the "topocentric" (i.e. in vacuo)
**  zenith distance.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**    phpa   double    pressure at the observer (hPa = millibar)
**    tc     double    ambient temperature at the observer (deg C)
**    rh     double    relative humidity at the observer (range 0-1)
**    wl     double    wavelength (micrometers)
**
**  Returned:
**    refa   double*   tan Z coefficient (radians)
**    refb   double*   tan^3 Z coefficient (radians)
**
**  Notes:
**
**  1) The model balances speed and accuracy to give good results in
**     applications where performance at low altitudes is not paramount.
**     Performance is maintained across a range of conditions, and
**     applies to both optical/IR and radio.
**
**  2) The model omits the effects of (i) height above sea level (apart
**     from the reduced pressure itself), (ii) latitude (i.e. the
**     flattening of the Earth), (iii) variations in tropospheric lapse
**     rate and (iv) dispersive effects in the radio.
**
**     The model was tested using the following range of conditions:
**
**       lapse rates 0.0055, 0.0065, 0.0075 deg/meter
**       latitudes 0, 25, 50, 75 degrees
**       heights 0, 2500, 5000 meters ASL
**       pressures mean for height -10% to +5% in steps of 5%
**       temperatures -10 deg to +20 deg with respect to 280 deg at SL
**       relative humidity 0, 0.5, 1
**       wavelengths 0.4, 0.6, ... 2 micron, + radio
**       zenith distances 15, 45, 75 degrees
**
**     The accuracy with respect to raytracing through a model
**     atmosphere was as follows:
**
**                            worst         RMS
**
**       optical/IR           62 mas       8 mas
**       radio               319 mas      49 mas
**
**     For this particular set of conditions:
**
**       lapse rate 0.0065 K/meter
**       latitude 50 degrees
**       sea level
**       pressure 1005 mb
**       temperature 280.15 K
**       humidity 80%
**       wavelength 5740 Angstroms
**
**     the results were as follows:
```

```
**
**       ZD       raytrace     iauRefco   Saastamoinen
**
**       10         10.27        10.27        10.27
**       20         21.19        21.20        21.19
**       30         33.61        33.61        33.60
**       40         48.82        48.83        48.81
**       45         58.16        58.18        58.16
**       50         69.28        69.30        69.27
**       55         82.97        82.99        82.95
**       60        100.51       100.54       100.50
**       65        124.23       124.26       124.20
**       70        158.63       158.68       158.61
**       72        177.32       177.37       177.31
**       74        200.35       200.38       200.32
**       76        229.45       229.43       229.42
**       78        267.44       267.29       267.41
**       80        319.13       318.55       319.10
**
**      deg        arcsec       arcsec       arcsec
**
**     The values for Saastamoinen's formula (which includes terms
**     up to tan^5) are taken from Hohenkerk and Sinclair (1985).
**
**  3) A wl value in the range 0-100 selects the optical/IR case and is
**     wavelength in micrometers.  Any value outside this range selects
**     the radio case.
**
**  4) Outlandish input parameters are silently limited to
**     mathematically safe values.  Zero pressure is permissible, and
**     causes zeroes to be returned.
**
**  5) The algorithm draws on several sources, as follows:
**
**     a) The formula for the saturation vapour pressure of water as
**        a function of temperature and temperature is taken from
**        Equations (A4.5-A4.7) of Gill (1982).
**
**     b) The formula for the water vapour pressure, given the
**        saturation pressure and the relative humidity, is from
**        Crane (1976), Equation (2.5.5).
**
**     c) The refractivity of air is a function of temperature,
**        total pressure, water-vapour pressure and, in the case
**        of optical/IR, wavelength.  The formulae for the two cases are
**        developed from Hohenkerk & Sinclair (1985) and Rueger (2002).
**        The IAG (1999) optical refractivity for dry air is used.
**
**     d) The formula for beta, the ratio of the scale height of the
**        atmosphere to the geocentric distance of the observer, is
**        an adaption of Equation (9) from Stone (1996).  The
**        adaptations, arrived at empirically, consist of (i) a small
**        adjustment to the coefficient and (ii) a humidity term for the
**        radio case only.
**
**     e) The formulae for the refraction constants as a function of
**        n-1 and beta are from Green (1987), Equation (4.31).
**
**  References:
**
**     Crane, R.K., Meeks, M.L. (ed), "Refraction Effects in the Neutral
**     Atmosphere", Methods of Experimental Physics: Astrophysics 12B,
**     Academic Press, 1976.
**
**     Gill, Adrian E., "Atmosphere-Ocean Dynamics", Academic Press,
**     1982.
**
**     Green, R.M., "Spherical Astronomy", Cambridge University Press,
**     1987.
**
**     Hohenkerk, C.Y., & Sinclair, A.T., NAO Technical Note No. 63,
**     1985.
**
```

```
**     IAG Resolutions adopted at the XXIIth General Assembly in
**     Birmingham, 1999, Resolution 3.
**
**     Rueger, J.M., "Refractive Index Formulae for Electronic Distance
**     Measurement with Radio and Millimetre Waves", in Unisurv Report
**     S-68, School of Surveying and Spatial Information Systems,
**     University of New South Wales, Sydney, Australia, 2002.
**
**     Stone, Ronald C., P.A.S.P. 108, 1051-1058, 1996.
**
*/
```

```
void iauRm2v(double r[3][3], double w[3])
/*
**  - - - - - - - -
**   i a u R m 2 v
**  - - - - - - - -
**
**  Express an r-matrix as an r-vector.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     r        double[3][3]    rotation matrix
**
**  Returned:
**     w        double[3]       rotation vector (Note 1)
**
**  Notes:
**
**  1) A rotation matrix describes a rotation through some angle about
**     some arbitrary axis called the Euler axis.  The "rotation vector"
**     returned by this function has the same direction as the Euler axis,
**     and its magnitude is the angle in radians.  (The magnitude and
**     direction can be separated by means of the function iauPn.)
**
**  2) If r is null, so is the result.  If r is not a rotation matrix
**     the result is undefined;  r must be proper (i.e. have a positive
**     determinant) and real orthogonal (inverse = transpose).
**
**  3) The reference frame rotates clockwise as seen looking along
**     the rotation vector from the origin.
**
*/
```

```
void iauRv2m(double w[3], double r[3][3])
/*
**  - - - - - - - -
**   i a u R v 2 m
**  - - - - - - - -
**
**  Form the r-matrix corresponding to a given r-vector.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     w        double[3]      rotation vector (Note 1)
**
**  Returned:
**     r        double[3][3]    rotation matrix
**
**  Notes:
**
**  1) A rotation matrix describes a rotation through some angle about
**     some arbitrary axis called the Euler axis.  The "rotation vector"
**     supplied to This function has the same direction as the Euler
**     axis, and its magnitude is the angle in radians.
**
**  2) If w is null, the identity matrix is returned.
**
**  3) The reference frame rotates clockwise as seen looking along the
**     rotation vector from the origin.
**
*/
```

```
void iauRx(double phi, double r[3][3])
/*
**  - - - - - -
**   i a u R x
**  - - - - - -
**
**  Rotate an r-matrix about the x-axis.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     phi    double          angle (radians)
**
**  Given and returned:
**     r      double[3][3]    r-matrix, rotated
**
**  Notes:
**
**  1) Calling this function with positive phi incorporates in the
**     supplied r-matrix r an additional rotation, about the x-axis,
**     anticlockwise as seen looking towards the origin from positive x.
**
**  2) The additional rotation can be represented by this matrix:
**
**         (  1        0            0      )
**         (                               )
**         (  0   + cos(phi)   + sin(phi)  )
**         (                               )
**         (  0   - sin(phi)   + cos(phi)  )
**
*/
```

```
void iauRxp(double r[3][3], double p[3], double rp[3])
/*
**  - - - - - - -
**   i a u R x p
**  - - - - - - -
**
**  Multiply a p-vector by an r-matrix.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     r        double[3][3]    r-matrix
**     p        double[3]       p-vector
**
**  Returned:
**     rp       double[3]       r * p
**
**  Note:
**     It is permissible for p and rp to be the same array.
**
**  Called:
**     iauCp        copy p-vector
**
*/
```

```
void iauRxpv(double r[3][3], double pv[2][3], double rpv[2][3])
/*
**  - - - - - - - - -
**   i a u R x p v
**  - - - - - - - - -
**
**  Multiply a pv-vector by an r-matrix.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     r        double[3][3]    r-matrix
**     pv       double[2][3]    pv-vector
**
**  Returned:
**     rpv      double[2][3]    r * pv
**
**  Notes:
**
**  1) The algorithm is for the simple case where the r-matrix r is not
**     a function of time.  The case where r is a function of time leads
**     to an additional velocity component equal to the product of the
**     derivative of r and the position vector.
**
**  2) It is permissible for pv and rpv to be the same array.
**
**  Called:
**     iauRxp       product of r-matrix and p-vector
**
*/
```

```
void iauRxr(double a[3][3], double b[3][3], double atb[3][3])
/*
**  - - - - - - -
**   i a u R x r
**  - - - - - - -
**
**  Multiply two r-matrices.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     a        double[3][3]    first r-matrix
**     b        double[3][3]    second r-matrix
**
**  Returned:
**     atb      double[3][3]    a * b
**
**  Note:
**     It is permissible to re-use the same array for any of the
**     arguments.
**
**  Called:
**     iauCr        copy r-matrix
**
*/
```

```
void iauRy(double theta, double r[3][3])
/*
**  - - - - - -
**   i a u R y
**  - - - - - -
**
**  Rotate an r-matrix about the y-axis.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     theta  double          angle (radians)
**
**  Given and returned:
**     r      double[3][3]    r-matrix, rotated
**
**  Notes:
**
**  1) Calling this function with positive theta incorporates in the
**     supplied r-matrix r an additional rotation, about the y-axis,
**     anticlockwise as seen looking towards the origin from positive y.
**
**  2) The additional rotation can be represented by this matrix:
**
**         (  + cos(theta)     0      - sin(theta)  )
**         (                                        )
**         (       0           1           0        )
**         (                                        )
**         (  + sin(theta)     0      + cos(theta)  )
**
*/
```

```
void iauRz(double psi, double r[3][3])
/*
**  - - - - - -
**   i a u R z
**  - - - - - -
**
**  Rotate an r-matrix about the z-axis.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     psi    double          angle (radians)
**
**  Given and returned:
**     r      double[3][3]    r-matrix, rotated
**
**  Notes:
**
**  1) Calling this function with positive psi incorporates in the
**     supplied r-matrix r an additional rotation, about the z-axis,
**     anticlockwise as seen looking towards the origin from positive z.
**
**  2) The additional rotation can be represented by this matrix:
**
**         (  + cos(psi)   + sin(psi)     0  )
**         (                                 )
**         (  - sin(psi)   + cos(psi)     0  )
**         (                                 )
**         (       0            0         1  )
**
*/
```

```
double iauS00(double date1, double date2, double x, double y)
/*
**  - - - - - - - -
**   i a u S 0 0
**  - - - - - - - -
**
**  The CIO locator s, positioning the Celestial Intermediate Origin on
**  the equator of the Celestial Intermediate Pole, given the CIP's X,Y
**  coordinates.  Compatible with IAU 2000A precession-nutation.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical model.
**
**  Given:
**     date1,date2   double    TT as a 2-part Julian Date (Note 1)
**     x,y           double    CIP coordinates (Note 3)
**
**  Returned (function value):
**                   double    the CIO locator s in radians (Note 2)
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The CIO locator s is the difference between the right ascensions
**     of the same point in two systems:  the two systems are the GCRS
**     and the CIP,CIO, and the point is the ascending node of the
**     CIP equator.  The quantity s remains below 0.1 arcsecond
**     throughout 1900-2100.
**
**  3) The series used to compute s is in fact for s+XY/2, where X and Y
**     are the x and y components of the CIP unit vector;  this series
**     is more compact than a direct series for s would be.  This
**     function requires X,Y to be supplied by the caller, who is
**     responsible for providing values that are consistent with the
**     supplied date.
**
**  4) The model is consistent with the IAU 2000A precession-nutation.
**
**  Called:
**     iauFal03     mean anomaly of the Moon
**     iauFalp03    mean anomaly of the Sun
**     iauFaf03     mean argument of the latitude of the Moon
**     iauFad03     mean elongation of the Moon from the Sun
**     iauFaom03    mean longitude of the Moon's ascending node
**     iauFave03    mean longitude of Venus
**     iauFae03     mean longitude of Earth
**     iauFapa03    general accumulated precession in longitude
**
**  References:
**
**     Capitaine, N., Chapront, J., Lambert, S. and Wallace, P.,
```

```
**     "Expressions for the Celestial Intermediate Pole and Celestial
**     Ephemeris Origin consistent with the IAU 2000A precession-
**     nutation model", Astron.Astrophys. 400, 1145-1154 (2003)
**
**     n.b. The celestial ephemeris origin (CEO) was renamed "celestial
**          intermediate origin" (CIO) by IAU 2006 Resolution 2.
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG (2004)
**
*/
```

```
double iauS00a(double date1, double date2)
/*
**  - - - - - - - -
**   i a u S 0 0 a
**  - - - - - - - -
**
**  The CIO locator s, positioning the Celestial Intermediate Origin on
**  the equator of the Celestial Intermediate Pole, using the IAU 2000A
**  precession-nutation model.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1,date2  double    TT as a 2-part Julian Date (Note 1)
**
**  Returned (function value):
**                  double    the CIO locator s in radians (Note 2)
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The CIO locator s is the difference between the right ascensions
**     of the same point in two systems.  The two systems are the GCRS
**     and the CIP,CIO, and the point is the ascending node of the
**     CIP equator.  The CIO locator s remains a small fraction of
**     1 arcsecond throughout 1900-2100.
**
**  3) The series used to compute s is in fact for s+XY/2, where X and Y
**     are the x and y components of the CIP unit vector;  this series
**     is more compact than a direct series for s would be.  The present
**     function uses the full IAU 2000A nutation model when predicting
**     the CIP position.  Faster results, with no significant loss of
**     accuracy, can be obtained via the function iauS00b, which uses
**     instead the IAU 2000B truncated model.
**
**  Called:
**     iauPnm00a    classical NPB matrix, IAU 2000A
**     iauBnp2xy    extract CIP X,Y from the BPN matrix
**     iauS00       the CIO locator s, given X,Y, IAU 2000A
**
**  References:
**
**     Capitaine, N., Chapront, J., Lambert, S. and Wallace, P.,
**     "Expressions for the Celestial Intermediate Pole and Celestial
**     Ephemeris Origin consistent with the IAU 2000A precession-
**     nutation model", Astron.Astrophys. 400, 1145-1154 (2003)
**
**     n.b. The celestial ephemeris origin (CEO) was renamed "celestial
**          intermediate origin" (CIO) by IAU 2006 Resolution 2.
**
```

```
**      McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**      IERS Technical Note No. 32, BKG (2004)
**
*/
```

```
double iauS00b(double date1, double date2)
/*
**  - - - - - - - -
**   i a u S 0 0 b
**  - - - - - - - -
**
**  The CIO locator s, positioning the Celestial Intermediate Origin on
**  the equator of the Celestial Intermediate Pole, using the IAU 2000B
**  precession-nutation model.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1,date2  double    TT as a 2-part Julian Date (Note 1)
**
**  Returned (function value):
**                  double    the CIO locator s in radians (Note 2)
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The CIO locator s is the difference between the right ascensions
**     of the same point in two systems.  The two systems are the GCRS
**     and the CIP,CIO, and the point is the ascending node of the
**     CIP equator.  The CIO locator s remains a small fraction of
**     1 arcsecond throughout 1900-2100.
**
**  3) The series used to compute s is in fact for s+XY/2, where X and Y
**     are the x and y components of the CIP unit vector;  this series
**     is more compact than a direct series for s would be.  The present
**     function uses the IAU 2000B truncated nutation model when
**     predicting the CIP position.  The function iauS00a uses instead
**     the full IAU 2000A model, but with no significant increase in
**     accuracy and at some cost in speed.
**
**  Called:
**     iauPnm00b    classical NPB matrix, IAU 2000B
**     iauBnp2xy    extract CIP X,Y from the BPN matrix
**     iauS00       the CIO locator s, given X,Y, IAU 2000A
**
**  References:
**
**     Capitaine, N., Chapront, J., Lambert, S. and Wallace, P.,
**     "Expressions for the Celestial Intermediate Pole and Celestial
**     Ephemeris Origin consistent with the IAU 2000A precession-
**     nutation model", Astron.Astrophys. 400, 1145-1154 (2003)
**
**     n.b. The celestial ephemeris origin (CEO) was renamed "celestial
**          intermediate origin" (CIO) by IAU 2006 Resolution 2.
**
```

```
**      McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**      IERS Technical Note No. 32, BKG (2004)
**
*/
```

```
double iauS06(double date1, double date2, double x, double y)
/*
**  - - - - - - -
**   i a u S 0 6
**  - - - - - - -
**
**  The CIO locator s, positioning the Celestial Intermediate Origin on
**  the equator of the Celestial Intermediate Pole, given the CIP's X,Y
**  coordinates.  Compatible with IAU 2006/2000A precession-nutation.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical model.
**
**  Given:
**     date1,date2   double    TT as a 2-part Julian Date (Note 1)
**     x,y           double    CIP coordinates (Note 3)
**
**  Returned (function value):
**                   double    the CIO locator s in radians (Note 2)
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The CIO locator s is the difference between the right ascensions
**     of the same point in two systems:  the two systems are the GCRS
**     and the CIP,CIO, and the point is the ascending node of the
**     CIP equator.  The quantity s remains below 0.1 arcsecond
**     throughout 1900-2100.
**
**  3) The series used to compute s is in fact for s+XY/2, where X and Y
**     are the x and y components of the CIP unit vector;  this series
**     is more compact than a direct series for s would be.  This
**     function requires X,Y to be supplied by the caller, who is
**     responsible for providing values that are consistent with the
**     supplied date.
**
**  4) The model is consistent with the "P03" precession (Capitaine et
**     al. 2003), adopted by IAU 2006 Resolution 1, 2006, and the
**     IAU 2000A nutation (with P03 adjustments).
**
**  Called:
**     iauFal03     mean anomaly of the Moon
**     iauFalp03    mean anomaly of the Sun
**     iauFaf03     mean argument of the latitude of the Moon
**     iauFad03     mean elongation of the Moon from the Sun
**     iauFaom03    mean longitude of the Moon's ascending node
**     iauFave03    mean longitude of Venus
**     iauFae03     mean longitude of Earth
**     iauFapa03    general accumulated precession in longitude
**
**  References:
```

```
**
**      Capitaine, N., Wallace, P.T. & Chapront, J., 2003, Astron.
**      Astrophys. 432, 355
**
**      McCarthy, D.D., Petit, G. (eds.) 2004, IERS Conventions (2003),
**      IERS Technical Note No. 32, BKG
**
*/
```

```
double iauS06a(double date1, double date2)
/*
**  - - - - - - - -
**   i a u S 0 6 a
**  - - - - - - - -
**
**  The CIO locator s, positioning the Celestial Intermediate Origin on
**  the equator of the Celestial Intermediate Pole, using the IAU 2006
**  precession and IAU 2000A nutation models.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1,date2  double    TT as a 2-part Julian Date (Note 1)
**
**  Returned (function value):
**                  double    the CIO locator s in radians (Note 2)
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The CIO locator s is the difference between the right ascensions
**     of the same point in two systems.  The two systems are the GCRS
**     and the CIP,CIO, and the point is the ascending node of the
**     CIP equator.  The CIO locator s remains a small fraction of
**     1 arcsecond throughout 1900-2100.
**
**  3) The series used to compute s is in fact for s+XY/2, where X and Y
**     are the x and y components of the CIP unit vector;  this series is
**     more compact than a direct series for s would be.  The present
**     function uses the full IAU 2000A nutation model when predicting
**     the CIP position.
**
**  Called:
**     iauPnm06a    classical NPB matrix, IAU 2006/2000A
**     iauBpn2xy    extract CIP X,Y coordinates from NPB matrix
**     iauS06       the CIO locator s, given X,Y, IAU 2006
**
**  References:
**
**     Capitaine, N., Chapront, J., Lambert, S. and Wallace, P.,
**     "Expressions for the Celestial Intermediate Pole and Celestial
**     Ephemeris Origin consistent with the IAU 2000A precession-
**     nutation model", Astron.Astrophys. 400, 1145-1154 (2003)
**
**     n.b. The celestial ephemeris origin (CEO) was renamed "celestial
**          intermediate origin" (CIO) by IAU 2006 Resolution 2.
**
**     Capitaine, N. & Wallace, P.T., 2006, Astron.Astrophys. 450, 855
**
```

```
**      McCarthy, D. D., Petit, G. (eds.), 2004, IERS Conventions (2003),
**      IERS Technical Note No. 32, BKG
**
**      Wallace, P.T. & Capitaine, N., 2006, Astron.Astrophys. 459, 981
**
*/
```

```
void iauS2c(double theta, double phi, double c[3])
/*
**  - - - - - - -
**   i a u S 2 c
**  - - - - - - -
**
**  Convert spherical coordinates to Cartesian.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     theta    double       longitude angle (radians)
**     phi      double       latitude angle (radians)
**
**  Returned:
**     c        double[3]    direction cosines
**
*/
```

```
void iauS2p(double theta, double phi, double r, double p[3])
/*
**  - - - - - - - -
**   i a u S 2 p
**  - - - - - - - -
**
**  Convert spherical polar coordinates to p-vector.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     theta   double       longitude angle (radians)
**     phi     double       latitude angle (radians)
**     r       double       radial distance
**
**  Returned:
**     p       double[3]    Cartesian coordinates
**
**  Called:
**     iauS2c       spherical coordinates to unit vector
**     iauSxp       multiply p-vector by scalar
**
*/
```

```
void iauS2pv(double theta, double phi, double r,
             double td, double pd, double rd,
             double pv[2][3])
/*
**  - - - - - - - -
**   i a u S 2 p v
**  - - - - - - - -
**
**  Convert position/velocity from spherical to Cartesian coordinates.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     theta    double          longitude angle (radians)
**     phi      double          latitude angle (radians)
**     r        double          radial distance
**     td       double          rate of change of theta
**     pd       double          rate of change of phi
**     rd       double          rate of change of r
**
**  Returned:
**     pv       double[2][3]    pv-vector
**
*/
```

```
void iauS2xpv(double s1, double s2, double pv[2][3], double spv[2][3])
/*
**  - - - - - - - - - -
**   i a u S 2 x p v
**  - - - - - - - - - -
**
**  Multiply a pv-vector by two scalars.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     s1     double         scalar to multiply position component by
**     s2     double         scalar to multiply velocity component by
**     pv     double[2][3]   pv-vector
**
**  Returned:
**     spv    double[2][3]   pv-vector: p scaled by s1, v scaled by s2
**
**  Note:
**     It is permissible for pv and spv to be the same array.
**
**  Called:
**     iauSxp       multiply p-vector by scalar
**
*/
```

```
double iauSepp(double a[3], double b[3])
/*
**  - - - - - - - -
**   i a u S e p p
**  - - - - - - - -
**
**  Angular separation between two p-vectors.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     a      double[3]    first p-vector (not necessarily unit length)
**     b      double[3]    second p-vector (not necessarily unit length)
**
**  Returned (function value):
**            double       angular separation (radians, always positive)
**
**  Notes:
**
**  1) If either vector is null, a zero result is returned.
**
**  2) The angular separation is most simply formulated in terms of
**     scalar product.  However, this gives poor accuracy for angles
**     near zero and pi.  The present algorithm uses both cross product
**     and dot product, to deliver full accuracy whatever the size of
**     the angle.
**
**  Called:
**     iauPxp       vector product of two p-vectors
**     iauPm        modulus of p-vector
**     iauPdp       scalar product of two p-vectors
**
*/
```

```
double iauSeps(double al, double ap, double bl, double bp)
/*
**  - - - - - - - -
**   i a u S e p s
**  - - - - - - - -
**
**  Angular separation between two sets of spherical coordinates.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     al     double       first longitude (radians)
**     ap     double       first latitude (radians)
**     bl     double       second longitude (radians)
**     bp     double       second latitude (radians)
**
**  Returned (function value):
**            double       angular separation (radians)
**
**  Called:
**     iauS2c       spherical coordinates to unit vector
**     iauSepp      angular separation between two p-vectors
**
*/
```

```
double iauSp00(double date1, double date2)
/*
**  - - - - - - - -
**   i a u S p 0 0
**  - - - - - - - -
**
**  The TIO locator s', positioning the Terrestrial Intermediate Origin
**  on the equator of the Celestial Intermediate Pole.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical model.
**
**  Given:
**     date1,date2  double    TT as a 2-part Julian Date (Note 1)
**
**  Returned (function value):
**                  double    the TIO locator s' in radians (Note 2)
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The TIO locator s' is obtained from polar motion observations by
**     numerical integration, and so is in essence unpredictable.
**     However, it is dominated by a secular drift of about
**     47 microarcseconds per century, which is the approximation
**     evaluated by the present function.
**
**  Reference:
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG (2004)
**
*/
```

```
int iauStarpm(double ra1, double dec1,
              double pmr1, double pmd1, double px1, double rv1,
              double ep1a, double ep1b, double ep2a, double ep2b,
              double *ra2, double *dec2,
              double *pmr2, double *pmd2, double *px2, double *rv2)
/*
**  - - - - - - - - - -
**   i a u S t a r p m
**  - - - - - - - - - -
**
**  Star proper motion:  update star catalog data for space motion.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     ra1    double     right ascension (radians), before
**     dec1   double     declination (radians), before
**     pmr1   double     RA proper motion (radians/year), before
**     pmd1   double     Dec proper motion (radians/year), before
**     px1    double     parallax (arcseconds), before
**     rv1    double     radial velocity (km/s, +ve = receding), before
**     ep1a   double     "before" epoch, part A (Note 1)
**     ep1b   double     "before" epoch, part B (Note 1)
**     ep2a   double     "after" epoch, part A (Note 1)
**     ep2b   double     "after" epoch, part B (Note 1)
**
**  Returned:
**     ra2    double     right ascension (radians), after
**     dec2   double     declination (radians), after
**     pmr2   double     RA proper motion (radians/year), after
**     pmd2   double     Dec proper motion (radians/year), after
**     px2    double     parallax (arcseconds), after
**     rv2    double     radial velocity (km/s, +ve = receding), after
**
**  Returned (function value):
**            int        status:
**                          -1 = system error (should not occur)
**                           0 = no warnings or errors
**                           1 = distance overridden (Note 6)
**                           2 = excessive velocity (Note 7)
**                           4 = solution didn't converge (Note 8)
**                        else = binary logical OR of the above warnings
**
**  Notes:
**
**  1) The starting and ending TDB dates ep1a+ep1b and ep2a+ep2b are
**     Julian Dates, apportioned in any convenient way between the two
**     parts (A and B).  For example, JD(TDB)=2450123.7 could be
**     expressed in any of these ways, among others:
**
**            epNa            epNb
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in cases
**     where the loss of several decimal digits of resolution is
**     acceptable.  The J2000 method is best matched to the way the
**     argument is handled internally and will deliver the optimum
**     resolution.  The MJD method and the date & time methods are both
**     good compromises between resolution and convenience.
**
**  2) In accordance with normal star-catalog conventions, the object's
**     right ascension and declination are freed from the effects of
**     secular aberration.  The frame, which is aligned to the catalog
**     equator and equinox, is Lorentzian and centered on the SSB.
```

```
**
**     The proper motions are the rate of change of the right ascension
**     and declination at the catalog epoch and are in radians per TDB
**     Julian year.
**
**     The parallax and radial velocity are in the same frame.
**
**  3) Care is needed with units.  The star coordinates are in radians
**     and the proper motions in radians per Julian year, but the
**     parallax is in arcseconds.
**
**  4) The RA proper motion is in terms of coordinate angle, not true
**     angle.  If the catalog uses arcseconds for both RA and Dec proper
**     motions, the RA proper motion will need to be divided by cos(Dec)
**     before use.
**
**  5) Straight-line motion at constant speed, in the inertial frame,
**     is assumed.
**
**  6) An extremely small (or zero or negative) parallax is interpreted
**     to mean that the object is on the "celestial sphere", the radius
**     of which is an arbitrary (large) value (see the iauStarpv
**     function for the value used).  When the distance is overridden in
**     this way, the status, initially zero, has 1 added to it.
**
**  7) If the space velocity is a significant fraction of c (see the
**     constant VMAX in the function iauStarpv), it is arbitrarily set
**     to zero.  When this action occurs, 2 is added to the status.
**
**  8) The relativistic adjustment carried out in the iauStarpv function
**     involves an iterative calculation.  If the process fails to
**     converge within a set number of iterations, 4 is added to the
**     status.
**
**  Called:
**     iauStarpv    star catalog data to space motion pv-vector
**     iauPvu       update a pv-vector
**     iauPdp       scalar product of two p-vectors
**     iauPvstar    space motion pv-vector to star catalog data
**
*/
```

```
int iauStarpv(double ra, double dec,
              double pmr, double pmd, double px, double rv,
              double pv[2][3])
/*
**  - - - - - - - - - -
**   i a u S t a r p v
**  - - - - - - - - - -
**
**  Convert star catalog coordinates to position+velocity vector.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given (Note 1):
**     ra     double        right ascension (radians)
**     dec    double        declination (radians)
**     pmr    double        RA proper motion (radians/year)
**     pmd    double        Dec proper motion (radians/year)
**     px     double        parallax (arcseconds)
**     rv     double        radial velocity (km/s, positive = receding)
**
**  Returned (Note 2):
**     pv     double[2][3]  pv-vector (au, au/day)
**
**  Returned (function value):
**            int           status:
**                              0 = no warnings
**                              1 = distance overridden (Note 6)
**                              2 = excessive speed (Note 7)
**                              4 = solution didn't converge (Note 8)
**                           else = binary logical OR of the above
**
**  Notes:
**
**  1) The star data accepted by this function are "observables" for an
**     imaginary observer at the solar-system barycenter.  Proper motion
**     and radial velocity are, strictly, in terms of barycentric
**     coordinate time, TCB.  For most practical applications, it is
**     permissible to neglect the distinction between TCB and ordinary
**     "proper" time on Earth (TT/TAI).  The result will, as a rule, be
**     limited by the intrinsic accuracy of the proper-motion and
**     radial-velocity data;  moreover, the pv-vector is likely to be
**     merely an intermediate result, so that a change of time unit
**     would cancel out overall.
**
**     In accordance with normal star-catalog conventions, the object's
**     right ascension and declination are freed from the effects of
**     secular aberration.  The frame, which is aligned to the catalog
**     equator and equinox, is Lorentzian and centered on the SSB.
**
**  2) The resulting position and velocity pv-vector is with respect to
**     the same frame and, like the catalog coordinates, is freed from
**     the effects of secular aberration.  Should the "coordinate
**     direction", where the object was located at the catalog epoch, be
**     required, it may be obtained by calculating the magnitude of the
**     position vector pv[0][0-2] dividing by the speed of light in
**     au/day to give the light-time, and then multiplying the space
**     velocity pv[1][0-2] by this light-time and adding the result to
**     pv[0][0-2].
**
**     Summarizing, the pv-vector returned is for most stars almost
**     identical to the result of applying the standard geometrical
**     "space motion" transformation.  The differences, which are the
**     subject of the Stumpff paper referenced below, are:
**
**     (i) In stars with significant radial velocity and proper motion,
**     the constantly changing light-time distorts the apparent proper
**     motion.  Note that this is a classical, not a relativistic,
**     effect.
```

```
**
**     (ii) The transformation complies with special relativity.
**
**  3) Care is needed with units.  The star coordinates are in radians
**     and the proper motions in radians per Julian year, but the
**     parallax is in arcseconds; the radial velocity is in km/s, but
**     the pv-vector result is in au and au/day.
**
**  4) The RA proper motion is in terms of coordinate angle, not true
**     angle.  If the catalog uses arcseconds for both RA and Dec proper
**     motions, the RA proper motion will need to be divided by cos(Dec)
**     before use.
**
**  5) Straight-line motion at constant speed, in the inertial frame,
**     is assumed.
**
**  6) An extremely small (or zero or negative) parallax is interpreted
**     to mean that the object is on the "celestial sphere", the radius
**     of which is an arbitrary (large) value (see the constant PXMIN).
**     When the distance is overridden in this way, the status,
**     initially zero, has 1 added to it.
**
**  7) If the space velocity is a significant fraction of c (see the
**     constant VMAX), it is arbitrarily set to zero.  When this action
**     occurs, 2 is added to the status.
**
**  8) The relativistic adjustment involves an iterative calculation.
**     If the process fails to converge within a set number (IMAX) of
**     iterations, 4 is added to the status.
**
**  9) The inverse transformation is performed by the function
**     iauPvstar.
**
**  Called:
**     iauS2pv      spherical coordinates to pv-vector
**     iauPm        modulus of p-vector
**     iauZp        zero p-vector
**     iauPn        decompose p-vector into modulus and direction
**     iauPdp       scalar product of two p-vectors
**     iauSxp       multiply p-vector by scalar
**     iauPmp       p-vector minus p-vector
**     iauPpp       p-vector plus p-vector
**
**  Reference:
**
**     Stumpff, P., 1985, Astron.Astrophys. 144, 232-240.
**
*/
```

```
void iauSxp(double s, double p[3], double sp[3])
/*
**  - - - - - - - -
**   i a u S x p
**  - - - - - - - -
**
**  Multiply a p-vector by a scalar.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     s      double        scalar
**     p      double[3]     p-vector
**
**  Returned:
**     sp     double[3]     s * p
**
**  Note:
**     It is permissible for p and sp to be the same array.
**
*/
```

```
void iauSxpv(double s, double pv[2][3], double spv[2][3])
/*
**  - - - - - - - -
**   i a u S x p v
**  - - - - - - - -
**
**  Multiply a pv-vector by a scalar.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     s       double           scalar
**     pv      double[2][3]     pv-vector
**
**  Returned:
**     spv     double[2][3]     s * pv
**
**  Note:
**     It is permissible for pv and spv to be the same array.
**
**  Called:
**     iauS2xpv     multiply pv-vector by two scalars
**
*/
```

```
int iauTaitt(double tai1, double tai2, double *tt1, double *tt2)
/*
**  - - - - - - - - -
**   i a u T a i t t
**  - - - - - - - - -
**
**  Time scale transformation:  International Atomic Time, TAI, to
**  Terrestrial Time, TT.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical.
**
**  Given:
**     tai1,tai2  double    TAI as a 2-part Julian Date
**
**  Returned:
**     tt1,tt2    double    TT as a 2-part Julian Date
**
**  Returned (function value):
**                int       status:  0 = OK
**
**  Note:
**
**     tai1+tai2 is Julian Date, apportioned in any convenient way
**     between the two arguments, for example where tai1 is the Julian
**     Day Number and tai2 is the fraction of a day.  The returned
**     tt1,tt2 follow suit.
**
**  References:
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG (2004)
**
**     Explanatory Supplement to the Astronomical Almanac,
**     P. Kenneth Seidelmann (ed), University Science Books (1992)
**
*/
```

```
int iauTaiut1(double tai1, double tai2, double dta,
              double *ut11, double *ut12)
/*
**  - - - - - - - - - -
**   i a u T a i u t 1
**  - - - - - - - - - -
**
**  Time scale transformation:  International Atomic Time, TAI, to
**  Universal Time, UT1.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical.
**
**  Given:
**     tai1,tai2  double    TAI as a 2-part Julian Date
**     dta        double    UT1-TAI in seconds
**
**  Returned:
**     ut11,ut12  double    UT1 as a 2-part Julian Date
**
**  Returned (function value):
**                int       status:  0 = OK
**
**  Notes:
**
**  1) tai1+tai2 is Julian Date, apportioned in any convenient way
**     between the two arguments, for example where tai1 is the Julian
**     Day Number and tai2 is the fraction of a day.  The returned
**     UT11,UT12 follow suit.
**
**  2) The argument dta, i.e. UT1-TAI, is an observed quantity, and is
**     available from IERS tabulations.
**
**  Reference:
**
**     Explanatory Supplement to the Astronomical Almanac,
**     P. Kenneth Seidelmann (ed), University Science Books (1992)
**
*/
```

```
int iauTaiutc(double tai1, double tai2, double *utc1, double *utc2)
/*
**  - - - - - - - - - -
**   i a u T a i u t c
**  - - - - - - - - - -
**
**  Time scale transformation:  International Atomic Time, TAI, to
**  Coordinated Universal Time, UTC.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical.
**
**  Given:
**     tai1,tai2  double   TAI as a 2-part Julian Date (Note 1)
**
**  Returned:
**     utc1,utc2  double   UTC as a 2-part quasi Julian Date (Notes 1-3)
**
**  Returned (function value):
**                int      status: +1 = dubious year (Note 4)
**                                  0 = OK
**                                 -1 = unacceptable date
**
**  Notes:
**
**  1) tai1+tai2 is Julian Date, apportioned in any convenient way
**     between the two arguments, for example where tai1 is the Julian
**     Day Number and tai2 is the fraction of a day.  The returned utc1
**     and utc2 form an analogous pair, except that a special convention
**     is used, to deal with the problem of leap seconds - see the next
**     note.
**
**  2) JD cannot unambiguously represent UTC during a leap second unless
**     special measures are taken.  The convention in the present
**     function is that the JD day represents UTC days whether the
**     length is 86399, 86400 or 86401 SI seconds.  In the 1960-1972 era
**     there were smaller jumps (in either direction) each time the
**     linear UTC(TAI) expression was changed, and these "mini-leaps"
**     are also included in the SOFA convention.
**
**  3) The function iauD2dtf can be used to transform the UTC quasi-JD
**     into calendar date and clock time, including UTC leap second
**     handling.
**
**  4) The warning status "dubious year" flags UTCs that predate the
**     introduction of the time scale or that are too far in the future
**     to be trusted.  See iauDat for further details.
**
**  Called:
**     iauUtctai    UTC to TAI
**
**  References:
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG (2004)
**
**     Explanatory Supplement to the Astronomical Almanac,
**     P. Kenneth Seidelmann (ed), University Science Books (1992)
**
*/
```

```
int iauTcbtdb(double tcb1, double tcb2, double *tdb1, double *tdb2)
/*
**  - - - - - - - - - -
**   i a u T c b t d b
**  - - - - - - - - - -
**
**  Time scale transformation:  Barycentric Coordinate Time, TCB, to
**  Barycentric Dynamical Time, TDB.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical.
**
**  Given:
**     tcb1,tcb2  double    TCB as a 2-part Julian Date
**
**  Returned:
**     tdb1,tdb2  double    TDB as a 2-part Julian Date
**
**  Returned (function value):
**                int       status:  0 = OK
**
**  Notes:
**
**  1) tcb1+tcb2 is Julian Date, apportioned in any convenient way
**     between the two arguments, for example where tcb1 is the Julian
**     Day Number and tcb2 is the fraction of a day.  The returned
**     tdb1,tdb2 follow suit.
**
**  2) The 2006 IAU General Assembly introduced a conventional linear
**     transformation between TDB and TCB.  This transformation
**     compensates for the drift between TCB and terrestrial time TT,
**     and keeps TDB approximately centered on TT.  Because the
**     relationship between TT and TCB depends on the adopted solar
**     system ephemeris, the degree of alignment between TDB and TT over
**     long intervals will vary according to which ephemeris is used.
**     Former definitions of TDB attempted to avoid this problem by
**     stipulating that TDB and TT should differ only by periodic
**     effects.  This is a good description of the nature of the
**     relationship but eluded precise mathematical formulation.  The
**     conventional linear relationship adopted in 2006 sidestepped
**     these difficulties whilst delivering a TDB that in practice was
**     consistent with values before that date.
**
**  3) TDB is essentially the same as Teph, the time argument for the
**     JPL solar system ephemerides.
**
**  Reference:
**
**     IAU 2006 Resolution B3
**
*/
```

```
int iauTcgtt(double tcg1, double tcg2, double *tt1, double *tt2)
/*
**  - - - - - - - - -
**   i a u T c g t t
**  - - - - - - - - -
**
**  Time scale transformation:  Geocentric Coordinate Time, TCG, to
**  Terrestrial Time, TT.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical.
**
**  Given:
**     tcg1,tcg2  double    TCG as a 2-part Julian Date
**
**  Returned:
**     tt1,tt2    double    TT as a 2-part Julian Date
**
**  Returned (function value):
**                int       status:  0 = OK
**
**  Note:
**
**     tcg1+tcg2 is Julian Date, apportioned in any convenient way
**     between the two arguments, for example where tcg1 is the Julian
**     Day Number and tcg22 is the fraction of a day.  The returned
**     tt1,tt2 follow suit.
**
**  References:
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG (2004)
**
**     IAU 2000 Resolution B1.9
**
*/
```

```
int iauTdbtcb(double tdb1, double tdb2, double *tcb1, double *tcb2)
/*
**  - - - - - - - - - -
**   i a u T d b t c b
**  - - - - - - - - - -
**
**  Time scale transformation:  Barycentric Dynamical Time, TDB, to
**  Barycentric Coordinate Time, TCB.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical.
**
**  Given:
**     tdb1,tdb2  double    TDB as a 2-part Julian Date
**
**  Returned:
**     tcb1,tcb2  double    TCB as a 2-part Julian Date
**
**  Returned (function value):
**                int       status:  0 = OK
**
**  Notes:
**
**  1) tdb1+tdb2 is Julian Date, apportioned in any convenient way
**     between the two arguments, for example where tdb1 is the Julian
**     Day Number and tdb2 is the fraction of a day.  The returned
**     tcb1,tcb2 follow suit.
**
**  2) The 2006 IAU General Assembly introduced a conventional linear
**     transformation between TDB and TCB.  This transformation
**     compensates for the drift between TCB and terrestrial time TT,
**     and keeps TDB approximately centered on TT.  Because the
**     relationship between TT and TCB depends on the adopted solar
**     system ephemeris, the degree of alignment between TDB and TT over
**     long intervals will vary according to which ephemeris is used.
**     Former definitions of TDB attempted to avoid this problem by
**     stipulating that TDB and TT should differ only by periodic
**     effects.  This is a good description of the nature of the
**     relationship but eluded precise mathematical formulation.  The
**     conventional linear relationship adopted in 2006 sidestepped
**     these difficulties whilst delivering a TDB that in practice was
**     consistent with values before that date.
**
**  3) TDB is essentially the same as Teph, the time argument for the
**     JPL solar system ephemerides.
**
**  Reference:
**
**     IAU 2006 Resolution B3
**
*/
```

```
int iauTdbtt(double tdb1, double tdb2, double dtr,
             double *tt1, double *tt2 )
/*
**  - - - - - - - - -
**   i a u T d b t t
**  - - - - - - - - -
**
**  Time scale transformation:  Barycentric Dynamical Time, TDB, to
**  Terrestrial Time, TT.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical.
**
**  Given:
**     tdb1,tdb2  double    TDB as a 2-part Julian Date
**     dtr        double    TDB-TT in seconds
**
**  Returned:
**     tt1,tt2    double    TT as a 2-part Julian Date
**
**  Returned (function value):
**                int       status:  0 = OK
**
**  Notes:
**
**  1) tdb1+tdb2 is Julian Date, apportioned in any convenient way
**     between the two arguments, for example where tdb1 is the Julian
**     Day Number and tdb2 is the fraction of a day.  The returned
**     tt1,tt2 follow suit.
**
**  2) The argument dtr represents the quasi-periodic component of the
**     GR transformation between TT and TCB.  It is dependent upon the
**     adopted solar-system ephemeris, and can be obtained by numerical
**     integration, by interrogating a precomputed time ephemeris or by
**     evaluating a model such as that implemented in the SOFA function
**     iauDtdb.   The quantity is dominated by an annual term of 1.7 ms
**     amplitude.
**
**  3) TDB is essentially the same as Teph, the time argument for the
**     JPL solar system ephemerides.
**
**  References:
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG (2004)
**
**     IAU 2006 Resolution 3
**
*/
```

```
int iauTf2a(char s, int ihour, int imin, double sec, double *rad)
/*
**  - - - - - - - -
**   i a u T f 2 a
**  - - - - - - - -
**
**  Convert hours, minutes, seconds to radians.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     s         char    sign:  '-' = negative, otherwise positive
**     ihour     int     hours
**     imin      int     minutes
**     sec       double  seconds
**
**  Returned:
**     rad       double  angle in radians
**
**  Returned (function value):
**               int     status:  0 = OK
**                                1 = ihour outside range 0-23
**                                2 = imin outside range 0-59
**                                3 = sec outside range 0-59.999...
**
**  Notes:
**
**  1)  The result is computed even if any of the range checks fail.
**
**  2)  Negative ihour, imin and/or sec produce a warning status, but
**      the absolute value is used in the conversion.
**
**  3)  If there are multiple errors, the status value reflects only the
**      first, the smallest taking precedence.
**
*/
```

```
int iauTf2d(char s, int ihour, int imin, double sec, double *days)
/*
**  - - - - - - - -
**   i a u T f 2 d
**  - - - - - - - -
**
**  Convert hours, minutes, seconds to days.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     s         char    sign:  '-' = negative, otherwise positive
**     ihour     int     hours
**     imin      int     minutes
**     sec       double  seconds
**
**  Returned:
**     days      double  interval in days
**
**  Returned (function value):
**               int     status:  0 = OK
**                                1 = ihour outside range 0-23
**                                2 = imin outside range 0-59
**                                3 = sec outside range 0-59.999...
**
**  Notes:
**
**  1)  The result is computed even if any of the range checks fail.
**
**  2)  Negative ihour, imin and/or sec produce a warning status, but
**      the absolute value is used in the conversion.
**
**  3)  If there are multiple errors, the status value reflects only the
**      first, the smallest taking precedence.
**
*/
```

```
int iauTpors(double xi, double eta, double a, double b,
             double *a01, double *b01, double *a02, double *b02)
/*
**  - - - - - - - - -
**   i a u T p o r s
**  - - - - - - - - -
**
**  In the tangent plane projection, given the rectangular coordinates
**  of a star and its spherical coordinates, determine the spherical
**  coordinates of the tangent point.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     xi,eta     double  rectangular coordinates of star image (Note 2)
**     a,b        double  star's spherical coordinates (Note 3)
**
**  Returned:
**     *a01,*b01  double  tangent point's spherical coordinates, Soln. 1
**     *a02,*b02  double  tangent point's spherical coordinates, Soln. 2
**
**  Returned (function value):
**                int     number of solutions:
**                        0 = no solutions returned (Note 5)
**                        1 = only the first solution is useful (Note 6)
**                        2 = both solutions are useful (Note 6)
**
**  Notes:
**
**  1) The tangent plane projection is also called the "gnomonic
**     projection" and the "central projection".
**
**  2) The eta axis points due north in the adopted coordinate system.
**     If the spherical coordinates are observed (RA,Dec), the tangent
**     plane coordinates (xi,eta) are conventionally called the
**     "standard coordinates".  If the spherical coordinates are with
**     respect to a right-handed triad, (xi,eta) are also right-handed.
**     The units of (xi,eta) are, effectively, radians at the tangent
**     point.
**
**  3) All angular arguments are in radians.
**
**  4) The angles a01 and a02 are returned in the range 0-2pi.  The
**     angles b01 and b02 are returned in the range +/-pi, but in the
**     usual, non-pole-crossing, case, the range is +/-pi/2.
**
**  5) Cases where there is no solution can arise only near the poles.
**     For example, it is clearly impossible for a star at the pole
**     itself to have a non-zero xi value, and hence it is meaningless
**     to ask where the tangent point would have to be to bring about
**     this combination of xi and dec.
**
**  6) Also near the poles, cases can arise where there are two useful
**     solutions.  The return value indicates whether the second of the
**     two solutions returned is useful;  1 indicates only one useful
**     solution, the usual case.
**
**  7) The basis of the algorithm is to solve the spherical triangle PSC,
**     where P is the north celestial pole, S is the star and C is the
**     tangent point.  The spherical coordinates of the tangent point are
**     [a0,b0];  writing rho^2 = (xi^2+eta^2) and r^2 = (1+rho^2), side c
**     is then (pi/2-b), side p is sqrt(xi^2+eta^2) and side s (to be
**     found) is (pi/2-b0).  Angle C is given by sin(C) = xi/rho and
**     cos(C) = eta/rho.  Angle P (to be found) is the longitude
**     difference between star and tangent point (a-a0).
**
**  8) This function is a member of the following set:
**
```

```
**         spherical      vector         solve for
**
**         iauTpxes      iauTpxev         xi,eta
**         iauTpsts      iauTpstv          star
**       > iauTpors <    iauTporv         origin
**
**  Called:
**     iauAnp       normalize angle into range 0 to 2pi
**
**  References:
**
**     Calabretta M.R. & Greisen, E.W., 2002, "Representations of
**     celestial coordinates in FITS", Astron.Astrophys. 395, 1077
**
**     Green, R.M., "Spherical Astronomy", Cambridge University Press,
**     1987, Chapter 13.
**
*/
```

```
int iauTporv(double xi, double eta, double v[3],
             double v01[3], double v02[3])
/*
**  - - - - - - - - -
**   i a u T p o r v
**  - - - - - - - - -
**
**  In the tangent plane projection, given the rectangular coordinates
**  of a star and its direction cosines, determine the direction
**  cosines of the tangent point.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     xi,eta   double    rectangular coordinates of star image (Note 2)
**     v        double[3] star's direction cosines (Note 3)
**
**  Returned:
**     v01      double[3] tangent point's direction cosines, Solution 1
**     v02      double[3] tangent point's direction cosines, Solution 2
**
**  Returned (function value):
**              int       number of solutions:
**                        0 = no solutions returned (Note 4)
**                        1 = only the first solution is useful (Note 5)
**                        2 = both solutions are useful (Note 5)
**
**  Notes:
**
**  1) The tangent plane projection is also called the "gnomonic
**     projection" and the "central projection".
**
**  2) The eta axis points due north in the adopted coordinate system.
**     If the direction cosines represent observed (RA,Dec), the tangent
**     plane coordinates (xi,eta) are conventionally called the
**     "standard coordinates".  If the direction cosines are with
**     respect to a right-handed triad, (xi,eta) are also right-handed.
**     The units of (xi,eta) are, effectively, radians at the tangent
**     point.
**
**  3) The vector v must be of unit length or the result will be wrong.
**
**  4) Cases where there is no solution can arise only near the poles.
**     For example, it is clearly impossible for a star at the pole
**     itself to have a non-zero xi value, and hence it is meaningless
**     to ask where the tangent point would have to be.
**
**  5) Also near the poles, cases can arise where there are two useful
**     solutions.  The return value indicates whether the second of the
**     two solutions returned is useful;  1 indicates only one useful
**     solution, the usual case.
**
**  6) The basis of the algorithm is to solve the spherical triangle
**     PSC, where P is the north celestial pole, S is the star and C is
**     the tangent point.  Calling the celestial spherical coordinates
**     of the star and tangent point (a,b) and (a0,b0) respectively, and
**     writing rho^2 = (xi^2+eta^2) and r^2 = (1+rho^2), and
**     transforming the vector v into (a,b) in the normal way, side c is
**     then (pi/2-b), side p is sqrt(xi^2+eta^2) and side s (to be
**     found) is (pi/2-b0), while angle C is given by sin(C) = xi/rho
**     and cos(C) = eta/rho;  angle P (to be found) is (a-a0).  After
**     solving the spherical triangle, the result (a0,b0) can be
**     expressed in vector form as v0.
**
**  7) This function is a member of the following set:
**
**         spherical      vector         solve for
**
```

```
**         iauTpxes      iauTpxev         xi,eta
**         iauTpsts      iauTpstv          star
**         iauTpors    > iauTporv <       origin
**
**  References:
**
**     Calabretta M.R. & Greisen, E.W., 2002, "Representations of
**     celestial coordinates in FITS", Astron.Astrophys. 395, 1077
**
**     Green, R.M., "Spherical Astronomy", Cambridge University Press,
**     1987, Chapter 13.
**
*/
```

```
void iauTpsts(double xi, double eta, double a0, double b0,
              double *a, double *b)
/*
**  - - - - - - - - -
**   i a u T p s t s
**  - - - - - - - - -
**
**  In the tangent plane projection, given the star's rectangular
**  coordinates and the spherical coordinates of the tangent point,
**  solve for the spherical coordinates of the star.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     xi,eta    double  rectangular coordinates of star image (Note 2)
**     a0,b0     double  tangent point's spherical coordinates
**
**  Returned:
**     *a,*b     double  star's spherical coordinates
**
**  1) The tangent plane projection is also called the "gnomonic
**     projection" and the "central projection".
**
**  2) The eta axis points due north in the adopted coordinate system.
**     If the spherical coordinates are observed (RA,Dec), the tangent
**     plane coordinates (xi,eta) are conventionally called the
**     "standard coordinates".  If the spherical coordinates are with
**     respect to a right-handed triad, (xi,eta) are also right-handed.
**     The units of (xi,eta) are, effectively, radians at the tangent
**     point.
**
**  3) All angular arguments are in radians.
**
**  4) This function is a member of the following set:
**
**         spherical      vector         solve for
**
**         iauTpxes      iauTpxev         xi,eta
**       > iauTpsts <    iauTpstv          star
**         iauTpors      iauTporv         origin
**
**  Called:
**     iauAnp       normalize angle into range 0 to 2pi
**
**  References:
**
**     Calabretta M.R. & Greisen, E.W., 2002, "Representations of
**     celestial coordinates in FITS", Astron.Astrophys. 395, 1077
**
**     Green, R.M., "Spherical Astronomy", Cambridge University Press,
**     1987, Chapter 13.
**
*/
```

```
void iauTpstv(double xi, double eta, double v0[3], double v[3])
/*
**  - - - - - - - - -
**   i a u T p s t v
**  - - - - - - - - -
**
**  In the tangent plane projection, given the star's rectangular
**  coordinates and the direction cosines of the tangent point, solve
**  for the direction cosines of the star.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     xi,eta  double     rectangular coordinates of star image (Note 2)
**     v0      double[3]  tangent point's direction cosines
**
**  Returned:
**     v       double[3]  star's direction cosines
**
**  1) The tangent plane projection is also called the "gnomonic
**     projection" and the "central projection".
**
**  2) The eta axis points due north in the adopted coordinate system.
**     If the direction cosines represent observed (RA,Dec), the tangent
**     plane coordinates (xi,eta) are conventionally called the
**     "standard coordinates".  If the direction cosines are with
**     respect to a right-handed triad, (xi,eta) are also right-handed.
**     The units of (xi,eta) are, effectively, radians at the tangent
**     point.
**
**  3) The method used is to complete the star vector in the (xi,eta)
**     based triad and normalize it, then rotate the triad to put the
**     tangent point at the pole with the x-axis aligned to zero
**     longitude.  Writing (a0,b0) for the celestial spherical
**     coordinates of the tangent point, the sequence of rotations is
**     (b-pi/2) around the x-axis followed by (-a-pi/2) around the
**     z-axis.
**
**  4) If vector v0 is not of unit length, the returned vector v will
**     be wrong.
**
**  5) If vector v0 points at a pole, the returned vector v will be
**     based on the arbitrary assumption that the longitude coordinate
**     of the tangent point is zero.
**
**  6) This function is a member of the following set:
**
**         spherical      vector         solve for
**
**         iauTpxes      iauTpxev         xi,eta
**         iauTpsts    > iauTpstv <        star
**         iauTpors      iauTporv         origin
**
**  References:
**
**     Calabretta M.R. & Greisen, E.W., 2002, "Representations of
**     celestial coordinates in FITS", Astron.Astrophys. 395, 1077
**
**     Green, R.M., "Spherical Astronomy", Cambridge University Press,
**     1987, Chapter 13.
**
*/
```

```
int iauTpxes(double a, double b, double a0, double b0,
             double *xi, double *eta)
/*
**  - - - - - - - - -
**   i a u T p x e s
**  - - - - - - - - -
**
**  In the tangent plane projection, given celestial spherical
**  coordinates for a star and the tangent point, solve for the star's
**  rectangular coordinates in the tangent plane.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     a,b       double  star's spherical coordinates
**     a0,b0     double  tangent point's spherical coordinates
**
**  Returned:
**     *xi,*eta  double  rectangular coordinates of star image (Note 2)
**
**  Returned (function value):
**               int     status:  0 = OK
**                                1 = star too far from axis
**                                2 = antistar on tangent plane
**                                3 = antistar too far from axis
**
**  Notes:
**
**  1) The tangent plane projection is also called the "gnomonic
**     projection" and the "central projection".
**
**  2) The eta axis points due north in the adopted coordinate system.
**     If the spherical coordinates are observed (RA,Dec), the tangent
**     plane coordinates (xi,eta) are conventionally called the
**     "standard coordinates".  For right-handed spherical coordinates,
**     (xi,eta) are also right-handed.  The units of (xi,eta) are,
**     effectively, radians at the tangent point.
**
**  3) All angular arguments are in radians.
**
**  4) This function is a member of the following set:
**
**         spherical      vector         solve for
**
**       > iauTpxes <    iauTpxev         xi,eta
**         iauTpsts      iauTpstv          star
**         iauTpors      iauTporv         origin
**
**  References:
**
**     Calabretta M.R. & Greisen, E.W., 2002, "Representations of
**     celestial coordinates in FITS", Astron.Astrophys. 395, 1077
**
**     Green, R.M., "Spherical Astronomy", Cambridge University Press,
**     1987, Chapter 13.
**
*/
```

```
int iauTpxev(double v[3], double v0[3], double *xi, double *eta)
/*
**  - - - - - - - - -
**   i a u T p x e v
**  - - - - - - - - -
**
**  In the tangent plane projection, given celestial direction cosines
**  for a star and the tangent point, solve for the star's rectangular
**  coordinates in the tangent plane.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     v         double[3]  direction cosines of star (Note 4)
**     v0        double[3]  direction cosines of tangent point (Note 4)
**
**  Returned:
**     *xi,*eta  double     tangent plane coordinates of star
**
**  Returned (function value):
**               int        status: 0 = OK
**                                  1 = star too far from axis
**                                  2 = antistar on tangent plane
**                                  3 = antistar too far from axis
**
**  Notes:
**
**  1) The tangent plane projection is also called the "gnomonic
**     projection" and the "central projection".
**
**  2) The eta axis points due north in the adopted coordinate system.
**     If the direction cosines represent observed (RA,Dec), the tangent
**     plane coordinates (xi,eta) are conventionally called the
**     "standard coordinates".  If the direction cosines are with
**     respect to a right-handed triad, (xi,eta) are also right-handed.
**     The units of (xi,eta) are, effectively, radians at the tangent
**     point.
**
**  3) The method used is to extend the star vector to the tangent
**     plane and then rotate the triad so that (x,y) becomes (xi,eta).
**     Writing (a,b) for the celestial spherical coordinates of the
**     star, the sequence of rotations is (a+pi/2) around the z-axis
**     followed by (pi/2-b) around the x-axis.
**
**  4) If vector v0 is not of unit length, or if vector v is of zero
**     length, the results will be wrong.
**
**  5) If v0 points at a pole, the returned (xi,eta) will be based on
**     the arbitrary assumption that the longitude coordinate of the
**     tangent point is zero.
**
**  6) This function is a member of the following set:
**
**         spherical      vector         solve for
**
**         iauTpxes    > iauTpxev <       xi,eta
**         iauTpsts      iauTpstv          star
**         iauTpors      iauTporv         origin
**
**  References:
**
**     Calabretta M.R. & Greisen, E.W., 2002, "Representations of
**     celestial coordinates in FITS", Astron.Astrophys. 395, 1077
**
**     Green, R.M., "Spherical Astronomy", Cambridge University Press,
**     1987, Chapter 13.
**
*/
```

```
void iauTr(double r[3][3], double rt[3][3])
/*
**  - - - - - -
**   i a u T r
**  - - - - - -
**
**  Transpose an r-matrix.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     r        double[3][3]    r-matrix
**
**  Returned:
**     rt       double[3][3]    transpose
**
**  Note:
**     It is permissible for r and rt to be the same array.
**
**  Called:
**     iauCr        copy r-matrix
**
*/
```

```
void iauTrxp(double r[3][3], double p[3], double trp[3])
/*
**  - - - - - - - -
**   i a u T r x p
**  - - - - - - - -
**
**  Multiply a p-vector by the transpose of an r-matrix.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     r        double[3][3]   r-matrix
**     p        double[3]      p-vector
**
**  Returned:
**     trp      double[3]      r^T * p
**
**  Note:
**     It is permissible for p and trp to be the same array.
**
**  Called:
**     iauTr        transpose r-matrix
**     iauRxp       product of r-matrix and p-vector
**
*/
```

```
void iauTrxpv(double r[3][3], double pv[2][3], double trpv[2][3])
/*
**  - - - - - - - - - -
**   i a u T r x p v
**  - - - - - - - - - -
**
**  Multiply a pv-vector by the transpose of an r-matrix.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Given:
**     r        double[3][3]    r-matrix
**     pv       double[2][3]    pv-vector
**
**  Returned:
**     trpv     double[2][3]    r^T * pv
**
**  Notes:
**
**  1) The algorithm is for the simple case where the r-matrix r is not
**     a function of time.  The case where r is a function of time leads
**     to an additional velocity component equal to the product of the
**     derivative of the transpose of r and the position vector.
**
**  2) It is permissible for pv and rpv to be the same array.
**
**  Called:
**     iauTr        transpose r-matrix
**     iauRxpv      product of r-matrix and pv-vector
**
*/
```

```
int iauTttai(double tt1, double tt2, double *tai1, double *tai2)
/*
**  - - - - - - - - -
**   i a u T t t a i
**  - - - - - - - - -
**
**  Time scale transformation:  Terrestrial Time, TT, to International
**  Atomic Time, TAI.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical.
**
**  Given:
**     tt1,tt2    double    TT as a 2-part Julian Date
**
**  Returned:
**     tai1,tai2  double    TAI as a 2-part Julian Date
**
**  Returned (function value):
**                int       status:  0 = OK
**
**  Note:
**
**     tt1+tt2 is Julian Date, apportioned in any convenient way between
**     the two arguments, for example where tt1 is the Julian Day Number
**     and tt2 is the fraction of a day.  The returned tai1,tai2 follow
**     suit.
**
**  References:
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG (2004)
**
**     Explanatory Supplement to the Astronomical Almanac,
**     P. Kenneth Seidelmann (ed), University Science Books (1992)
**
*/
```

```
int iauTttcg(double tt1, double tt2, double *tcg1, double *tcg2)
/*
**  - - - - - - - - - -
**   i a u T t t c g
**  - - - - - - - - - -
**
**  Time scale transformation:  Terrestrial Time, TT, to Geocentric
**  Coordinate Time, TCG.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical.
**
**  Given:
**     tt1,tt2    double    TT as a 2-part Julian Date
**
**  Returned:
**     tcg1,tcg2  double    TCG as a 2-part Julian Date
**
**  Returned (function value):
**                int       status:  0 = OK
**
**  Note:
**
**     tt1+tt2 is Julian Date, apportioned in any convenient way between
**     the two arguments, for example where tt1 is the Julian Day Number
**     and tt2 is the fraction of a day.  The returned tcg1,tcg2 follow
**     suit.
**
**  References:
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG (2004)
**
**     IAU 2000 Resolution B1.9
**
*/
```

```
int iauTttdb(double tt1, double tt2, double dtr,
             double *tdb1, double *tdb2)
/*
**  - - - - - - - - -
**   i a u T t t d b
**  - - - - - - - - -
**
**  Time scale transformation:  Terrestrial Time, TT, to Barycentric
**  Dynamical Time, TDB.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical.
**
**  Given:
**     tt1,tt2    double    TT as a 2-part Julian Date
**     dtr        double    TDB-TT in seconds
**
**  Returned:
**     tdb1,tdb2  double    TDB as a 2-part Julian Date
**
**  Returned (function value):
**                int       status:  0 = OK
**
**  Notes:
**
**  1) tt1+tt2 is Julian Date, apportioned in any convenient way between
**     the two arguments, for example where tt1 is the Julian Day Number
**     and tt2 is the fraction of a day.  The returned tdb1,tdb2 follow
**     suit.
**
**  2) The argument dtr represents the quasi-periodic component of the
**     GR transformation between TT and TCB.  It is dependent upon the
**     adopted solar-system ephemeris, and can be obtained by numerical
**     integration, by interrogating a precomputed time ephemeris or by
**     evaluating a model such as that implemented in the SOFA function
**     iauDtdb.   The quantity is dominated by an annual term of 1.7 ms
**     amplitude.
**
**  3) TDB is essentially the same as Teph, the time argument for the JPL
**     solar system ephemerides.
**
**  References:
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG (2004)
**
**     IAU 2006 Resolution 3
**
*/
```

```c
int iauTtut1(double tt1, double tt2, double dt,
             double *ut11, double *ut12)
/*
**  - - - - - - - - -
**   i a u T t u t 1
**  - - - - - - - - -
**
**  Time scale transformation:  Terrestrial Time, TT, to Universal Time,
**  UT1.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical.
**
**  Given:
**     tt1,tt2    double    TT as a 2-part Julian Date
**     dt         double    TT-UT1 in seconds
**
**  Returned:
**     ut11,ut12  double    UT1 as a 2-part Julian Date
**
**  Returned (function value):
**                int       status:  0 = OK
**
**  Notes:
**
**  1) tt1+tt2 is Julian Date, apportioned in any convenient way between
**     the two arguments, for example where tt1 is the Julian Day Number
**     and tt2 is the fraction of a day.  The returned ut11,ut12 follow
**     suit.
**
**  2) The argument dt is classical Delta T.
**
**  Reference:
**
**     Explanatory Supplement to the Astronomical Almanac,
**     P. Kenneth Seidelmann (ed), University Science Books (1992)
**
*/
```

```
int iauUt1tai(double ut11, double ut12, double dta,
              double *tai1, double *tai2)
/*
**  - - - - - - - - - -
**   i a u U t 1 t a i
**  - - - - - - - - - -
**
**  Time scale transformation:  Universal Time, UT1, to International
**  Atomic Time, TAI.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical.
**
**  Given:
**     ut11,ut12  double    UT1 as a 2-part Julian Date
**     dta        double    UT1-TAI in seconds
**
**  Returned:
**     tai1,tai2  double    TAI as a 2-part Julian Date
**
**  Returned (function value):
**                int       status:  0 = OK
**
**  Notes:
**
**  1) ut11+ut12 is Julian Date, apportioned in any convenient way
**     between the two arguments, for example where ut11 is the Julian
**     Day Number and ut12 is the fraction of a day.  The returned
**     tai1,tai2 follow suit.
**
**  2) The argument dta, i.e. UT1-TAI, is an observed quantity, and is
**     available from IERS tabulations.
**
**  Reference:
**
**     Explanatory Supplement to the Astronomical Almanac,
**     P. Kenneth Seidelmann (ed), University Science Books (1992)
**
*/
```

```
int iauUt1tt(double ut11, double ut12, double dt,
             double *tt1, double *tt2)
/*
**  - - - - - - - - -
**   i a u U t 1 t t
**  - - - - - - - - -
**
**  Time scale transformation:  Universal Time, UT1, to Terrestrial
**  Time, TT.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical.
**
**  Given:
**     ut11,ut12  double    UT1 as a 2-part Julian Date
**     dt         double    TT-UT1 in seconds
**
**  Returned:
**     tt1,tt2    double    TT as a 2-part Julian Date
**
**  Returned (function value):
**                int       status:  0 = OK
**
**  Notes:
**
**  1) ut11+ut12 is Julian Date, apportioned in any convenient way
**     between the two arguments, for example where ut11 is the Julian
**     Day Number and ut12 is the fraction of a day.  The returned
**     tt1,tt2 follow suit.
**
**  2) The argument dt is classical Delta T.
**
**  Reference:
**
**     Explanatory Supplement to the Astronomical Almanac,
**     P. Kenneth Seidelmann (ed), University Science Books (1992)
**
*/
```

```
int iauUt1utc(double ut11, double ut12, double dut1,
              double *utc1, double *utc2)
/*
**  - - - - - - - - - -
**   i a u U t 1 u t c
**  - - - - - - - - - -
**
**  Time scale transformation:  Universal Time, UT1, to Coordinated
**  Universal Time, UTC.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical.
**
**  Given:
**     ut11,ut12  double   UT1 as a 2-part Julian Date (Note 1)
**     dut1       double   Delta UT1: UT1-UTC in seconds (Note 2)
**
**  Returned:
**     utc1,utc2  double   UTC as a 2-part quasi Julian Date (Notes 3,4)
**
**  Returned (function value):
**                int      status: +1 = dubious year (Note 5)
**                                  0 = OK
**                                 -1 = unacceptable date
**
**  Notes:
**
**  1) ut11+ut12 is Julian Date, apportioned in any convenient way
**     between the two arguments, for example where ut11 is the Julian
**     Day Number and ut12 is the fraction of a day.  The returned utc1
**     and utc2 form an analogous pair, except that a special convention
**     is used, to deal with the problem of leap seconds - see Note 3.
**
**  2) Delta UT1 can be obtained from tabulations provided by the
**     International Earth Rotation and Reference Systems Service.  The
**     value changes abruptly by 1s at a leap second;  however, close to
**     a leap second the algorithm used here is tolerant of the "wrong"
**     choice of value being made.
**
**  3) JD cannot unambiguously represent UTC during a leap second unless
**     special measures are taken.  The convention in the present
**     function is that the returned quasi-JD UTC1+UTC2 represents UTC
**     days whether the length is 86399, 86400 or 86401 SI seconds.
**
**  4) The function iauD2dtf can be used to transform the UTC quasi-JD
**     into calendar date and clock time, including UTC leap second
**     handling.
**
**  5) The warning status "dubious year" flags UTCs that predate the
**     introduction of the time scale or that are too far in the future
**     to be trusted.  See iauDat for further details.
**
**  Called:
**     iauJd2cal    JD to Gregorian calendar
**     iauDat       delta(AT) = TAI-UTC
**     iauCal2jd    Gregorian calendar to JD
**
**  References:
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG (2004)
**
**     Explanatory Supplement to the Astronomical Almanac,
**     P. Kenneth Seidelmann (ed), University Science Books (1992)
**
*/
```

```
int iauUtctai(double utc1, double utc2, double *tai1, double *tai2)
/*
**  - - - - - - - - - -
**   i a u U t c t a i
**  - - - - - - - - - -
**
**  Time scale transformation:  Coordinated Universal Time, UTC, to
**  International Atomic Time, TAI.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical.
**
**  Given:
**     utc1,utc2  double   UTC as a 2-part quasi Julian Date (Notes 1-4)
**
**  Returned:
**     tai1,tai2  double   TAI as a 2-part Julian Date (Note 5)
**
**  Returned (function value):
**                int      status: +1 = dubious year (Note 3)
**                                  0 = OK
**                                 -1 = unacceptable date
**
**  Notes:
**
**  1) utc1+utc2 is quasi Julian Date (see Note 2), apportioned in any
**     convenient way between the two arguments, for example where utc1
**     is the Julian Day Number and utc2 is the fraction of a day.
**
**  2) JD cannot unambiguously represent UTC during a leap second unless
**     special measures are taken.  The convention in the present
**     function is that the JD day represents UTC days whether the
**     length is 86399, 86400 or 86401 SI seconds.  In the 1960-1972 era
**     there were smaller jumps (in either direction) each time the
**     linear UTC(TAI) expression was changed, and these "mini-leaps"
**     are also included in the SOFA convention.
**
**  3) The warning status "dubious year" flags UTCs that predate the
**     introduction of the time scale or that are too far in the future
**     to be trusted.  See iauDat for further details.
**
**  4) The function iauDtf2d converts from calendar date and time of day
**     into 2-part Julian Date, and in the case of UTC implements the
**     leap-second-ambiguity convention described above.
**
**  5) The returned TAI1,TAI2 are such that their sum is the TAI Julian
**     Date.
**
**  Called:
**     iauJd2cal    JD to Gregorian calendar
**     iauDat       delta(AT) = TAI-UTC
**     iauCal2jd    Gregorian calendar to JD
**
**  References:
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG (2004)
**
**     Explanatory Supplement to the Astronomical Almanac,
**     P. Kenneth Seidelmann (ed), University Science Books (1992)
**
*/
```

```
int iauUtcut1(double utc1, double utc2, double dut1,
              double *ut11, double *ut12)
/*
**  - - - - - - - - - -
**   i a u U t c u t 1
**  - - - - - - - - - -
**
**  Time scale transformation:  Coordinated Universal Time, UTC, to
**  Universal Time, UT1.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical.
**
**  Given:
**     utc1,utc2  double   UTC as a 2-part quasi Julian Date (Notes 1-4)
**     dut1       double   Delta UT1 = UT1-UTC in seconds (Note 5)
**
**  Returned:
**     ut11,ut12  double   UT1 as a 2-part Julian Date (Note 6)
**
**  Returned (function value):
**                int      status: +1 = dubious year (Note 3)
**                                  0 = OK
**                                 -1 = unacceptable date
**
**  Notes:
**
**  1) utc1+utc2 is quasi Julian Date (see Note 2), apportioned in any
**     convenient way between the two arguments, for example where utc1
**     is the Julian Day Number and utc2 is the fraction of a day.
**
**  2) JD cannot unambiguously represent UTC during a leap second unless
**     special measures are taken.  The convention in the present
**     function is that the JD day represents UTC days whether the
**     length is 86399, 86400 or 86401 SI seconds.
**
**  3) The warning status "dubious year" flags UTCs that predate the
**     introduction of the time scale or that are too far in the future
**     to be trusted.  See iauDat for further details.
**
**  4) The function iauDtf2d converts from calendar date and time of
**     day into 2-part Julian Date, and in the case of UTC implements
**     the leap-second-ambiguity convention described above.
**
**  5) Delta UT1 can be obtained from tabulations provided by the
**     International Earth Rotation and Reference Systems Service.
**     It is the caller's responsibility to supply a dut1 argument
**     containing the UT1-UTC value that matches the given UTC.
**
**  6) The returned ut11,ut12 are such that their sum is the UT1 Julian
**     Date.
**
**  References:
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG (2004)
**
**     Explanatory Supplement to the Astronomical Almanac,
**     P. Kenneth Seidelmann (ed), University Science Books (1992)
**
**  Called:
**     iauJd2cal    JD to Gregorian calendar
**     iauDat       delta(AT) = TAI-UTC
**     iauUtctai    UTC to TAI
**     iauTaiut1    TAI to UT1
**
*/
```

```
void iauXy06(double date1, double date2, double *x, double *y)
/*
**  - - - - - - - -
**   i a u X y 0 6
**  - - - - - - - -
**
**  X,Y coordinates of celestial intermediate pole from series based
**  on IAU 2006 precession and IAU 2000A nutation.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  canonical model.
**
**  Given:
**     date1,date2  double     TT as a 2-part Julian Date (Note 1)
**
**  Returned:
**     x,y          double     CIP X,Y coordinates (Note 2)
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The X,Y coordinates are those of the unit vector towards the
**     celestial intermediate pole.  They represent the combined effects
**     of frame bias, precession and nutation.
**
**  3) The fundamental arguments used are as adopted in IERS Conventions
**     (2003) and are from Simon et al. (1994) and Souchay et al.
**     (1999).
**
**  4) This is an alternative to the angles-based method, via the SOFA
**     function iauFw2xy and as used in iauXys06a for example.  The two
**     methods agree at the 1 microarcsecond level (at present), a
**     negligible amount compared with the intrinsic accuracy of the
**     models.  However, it would be unwise to mix the two methods
**     (angles-based and series-based) in a single application.
**
**  Called:
**     iauFal03     mean anomaly of the Moon
**     iauFalp03    mean anomaly of the Sun
**     iauFaf03     mean argument of the latitude of the Moon
**     iauFad03     mean elongation of the Moon from the Sun
**     iauFaom03    mean longitude of the Moon's ascending node
**     iauFame03    mean longitude of Mercury
**     iauFave03    mean longitude of Venus
**     iauFae03     mean longitude of Earth
**     iauFama03    mean longitude of Mars
**     iauFaju03    mean longitude of Jupiter
**     iauFasa03    mean longitude of Saturn
**     iauFaur03    mean longitude of Uranus
**     iauFane03    mean longitude of Neptune
**     iauFapa03    general accumulated precession in longitude
```

```
**
**  References:
**
**     Capitaine, N., Wallace, P.T. & Chapront, J., 2003,
**     Astron.Astrophys., 412, 567
**
**     Capitaine, N. & Wallace, P.T., 2006, Astron.Astrophys. 450, 855
**
**     McCarthy, D. D., Petit, G. (eds.), 2004, IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG
**
**     Simon, J.L., Bretagnon, P., Chapront, J., Chapront-Touze, M.,
**     Francou, G. & Laskar, J., Astron.Astrophys., 1994, 282, 663
**
**     Souchay, J., Loysel, B., Kinoshita, H., Folgueira, M., 1999,
**     Astron.Astrophys.Supp.Ser. 135, 111
**
**     Wallace, P.T. & Capitaine, N., 2006, Astron.Astrophys. 459, 981
**
*/
```

```
void iauXys00a(double date1, double date2,
               double *x, double *y, double *s)
/*
**  - - - - - - - - - -
**   i a u X y s 0 0 a
**  - - - - - - - - - -
**
**  For a given TT date, compute the X,Y coordinates of the Celestial
**  Intermediate Pole and the CIO locator s, using the IAU 2000A
**  precession-nutation model.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1,date2  double   TT as a 2-part Julian Date (Note 1)
**
**  Returned:
**     x,y          double   Celestial Intermediate Pole (Note 2)
**     s            double   the CIO locator s (Note 3)
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The Celestial Intermediate Pole coordinates are the x,y
**     components of the unit vector in the Geocentric Celestial
**     Reference System.
**
**  3) The CIO locator s (in radians) positions the Celestial
**     Intermediate Origin on the equator of the CIP.
**
**  4) A faster, but slightly less accurate result (about 1 mas for
**     X,Y), can be obtained by using instead the iauXys00b function.
**
**  Called:
**     iauPnm00a    classical NPB matrix, IAU 2000A
**     iauBpn2xy    extract CIP X,Y coordinates from NPB matrix
**     iauS00       the CIO locator s, given X,Y, IAU 2000A
**
**  Reference:
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG (2004)
**
*/
```

```
void iauXys00b(double date1, double date2,
               double *x, double *y, double *s)
/*
**  - - - - - - - - - - -
**   i a u X y s 0 0 b
**  - - - - - - - - - - -
**
**  For a given TT date, compute the X,Y coordinates of the Celestial
**  Intermediate Pole and the CIO locator s, using the IAU 2000B
**  precession-nutation model.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1,date2  double   TT as a 2-part Julian Date (Note 1)
**
**  Returned:
**     x,y          double   Celestial Intermediate Pole (Note 2)
**     s            double   the CIO locator s (Note 3)
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The Celestial Intermediate Pole coordinates are the x,y
**     components of the unit vector in the Geocentric Celestial
**     Reference System.
**
**  3) The CIO locator s (in radians) positions the Celestial
**     Intermediate Origin on the equator of the CIP.
**
**  4) The present function is faster, but slightly less accurate (about
**     1 mas in X,Y), than the iauXys00a function.
**
**  Called:
**     iauPnm00b    classical NPB matrix, IAU 2000B
**     iauBpn2xy    extract CIP X,Y coordinates from NPB matrix
**     iauS00       the CIO locator s, given X,Y, IAU 2000A
**
**  Reference:
**
**     McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003),
**     IERS Technical Note No. 32, BKG (2004)
**
*/
```

```
void iauXys06a(double date1, double date2,
               double *x, double *y, double *s)
/*
**  - - - - - - - - - -
**   i a u X y s 0 6 a
**  - - - - - - - - - -
**
**  For a given TT date, compute the X,Y coordinates of the Celestial
**  Intermediate Pole and the CIO locator s, using the IAU 2006
**  precession and IAU 2000A nutation models.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  support function.
**
**  Given:
**     date1,date2  double  TT as a 2-part Julian Date (Note 1)
**
**  Returned:
**     x,y          double  Celestial Intermediate Pole (Note 2)
**     s            double  the CIO locator s (Note 3)
**
**  Notes:
**
**  1) The TT date date1+date2 is a Julian Date, apportioned in any
**     convenient way between the two arguments.  For example,
**     JD(TT)=2450123.7 could be expressed in any of these ways,
**     among others:
**
**            date1          date2
**
**         2450123.7           0.0       (JD method)
**         2451545.0       -1421.3       (J2000 method)
**         2400000.5       50123.2       (MJD method)
**         2450123.5           0.2       (date & time method)
**
**     The JD method is the most natural and convenient to use in
**     cases where the loss of several decimal digits of resolution
**     is acceptable.  The J2000 method is best matched to the way
**     the argument is handled internally and will deliver the
**     optimum resolution.  The MJD method and the date & time methods
**     are both good compromises between resolution and convenience.
**
**  2) The Celestial Intermediate Pole coordinates are the x,y components
**     of the unit vector in the Geocentric Celestial Reference System.
**
**  3) The CIO locator s (in radians) positions the Celestial
**     Intermediate Origin on the equator of the CIP.
**
**  4) Series-based solutions for generating X and Y are also available:
**     see Capitaine & Wallace (2006) and iauXy06.
**
**  Called:
**     iauPnm06a    classical NPB matrix, IAU 2006/2000A
**     iauBpn2xy    extract CIP X,Y coordinates from NPB matrix
**     iauS06       the CIO locator s, given X,Y, IAU 2006
**
**  References:
**
**     Capitaine, N. & Wallace, P.T., 2006, Astron.Astrophys. 450, 855
**
**     Wallace, P.T. & Capitaine, N., 2006, Astron.Astrophys. 459, 981
**
*/
```

```
void iauZp(double p[3])
/*
**  - - - - - -
**   i a u Z p
**  - - - - - -
**
**  Zero a p-vector.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Returned:
**     p        double[3]      zero p-vector
**
*/
```

```
void iauZpv(double pv[2][3])
/*
**  - - - - - - -
**   i a u Z p v
**  - - - - - - -
**
**  Zero a pv-vector.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Returned:
**     pv       double[2][3]      zero pv-vector
**
**  Called:
**     iauZp        zero p-vector
**
*/
```

```
void iauZr(double r[3][3])
/*
**  - - - - - -
**   i a u Z r
**  - - - - - -
**
**  Initialize an r-matrix to the null matrix.
**
**  This function is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Status:  vector/matrix support function.
**
**  Returned:
**     r        double[3][3]    r-matrix
**
*/
```

COPYRIGHT NOTICE

Text equivalent to that below appears at the end of every SOFA routine (with one exception). There are small formatting differences between the Fortran and C versions.

The one exception is the "leap second" routine DAT. This uniquely is classified as "user replaceable", and has a mitigated license statement that permits the distribution of local variants under the same name. This measure allows other SOFA routines to call the local variant, which may be file or network based, or otherwise equipped to pick up IERS leap second updates with no need to download new SOFA code.

```
*+----------------------------------------------------------------------
*
*  Copyright (C) 2023
*  Standards of Fundamental Astronomy Board
*  of the International Astronomical Union.
*
*  =====================
*  SOFA Software License
*  =====================
*
*  NOTICE TO USER:
*
*  BY USING THIS SOFTWARE YOU ACCEPT THE FOLLOWING SIX TERMS AND
*  CONDITIONS WHICH APPLY TO ITS USE.
*
*  1. The Software is owned by the IAU SOFA Board ("SOFA").
*
*  2. Permission is granted to anyone to use the SOFA software for any
*     purpose, including commercial applications, free of charge and
*     without payment of royalties, subject to the conditions and
*     restrictions listed below.
*
*  3. You (the user) may copy and distribute SOFA source code to others,
*     and use and adapt its code and algorithms in your own software,
*     on a world-wide, royalty-free basis.  That portion of your
*     distribution that does not consist of intact and unchanged copies
*     of SOFA source code files is a "derived work" that must comply
*     with the following requirements:
*
*     a) Your work shall be marked or carry a statement that it
*        (i) uses routines and computations derived by you from
*        software provided by SOFA under license to you; and
*        (ii) does not itself constitute software provided by and/or
*        endorsed by SOFA.
*
*     b) The source code of your derived work must contain descriptions
*        of how the derived work is based upon, contains and/or differs
*        from the original SOFA software.
*
*     c) The names of all routines in your derived work shall not
*        include the prefix "iau" or "sofa" or trivial modifications
*        thereof such as changes of case.
*
*     d) The origin of the SOFA components of your derived work must
*        not be misrepresented;  you must not claim that you wrote the
*        original software, nor file a patent application for SOFA
*        software or algorithms embedded in the SOFA software.
*
*     e) These requirements must be reproduced intact in any source
*        distribution and shall apply to anyone to whom you have
*        granted a further right to modify the source code of your
*        derived work.
*
*     Note that, as originally distributed, the SOFA software is
*     intended to be a definitive implementation of the IAU standards,
*     and consequently third-party modifications are discouraged.  All
*     variations, no matter how minor, must be explicitly marked as
```

SOFA Fortran constants

These must be used exactly as presented below.

```
*  Pi
      DOUBLE PRECISION DPI
      PARAMETER ( DPI = 3.141592653589793238462643D0 )

*  2Pi
      DOUBLE PRECISION D2PI
      PARAMETER ( D2PI = 6.283185307179586476925287D0 )

*  Radians to hours
      DOUBLE PRECISION DR2H
      PARAMETER ( DR2H = 3.819718634205488058453210D0 )

*  Radians to seconds
      DOUBLE PRECISION DR2S
      PARAMETER ( DR2S = 13750.98708313975701043156D0 )

*  Radians to degrees
      DOUBLE PRECISION DR2D
      PARAMETER ( DR2D = 57.29577951308232087679815D0 )

*  Radians to arc seconds
      DOUBLE PRECISION DR2AS
      PARAMETER ( DR2AS = 206264.8062470963551564734D0 )

*  Hours to radians
      DOUBLE PRECISION DH2R
      PARAMETER ( DH2R = 0.2617993877991494365385536D0 )

*  Seconds to radians
      DOUBLE PRECISION DS2R
      PARAMETER ( DS2R = 7.272205216643039903848712D-5 )

*  Degrees to radians
      DOUBLE PRECISION DD2R
      PARAMETER ( DD2R = 1.745329251994329576923691D-2 )

*  Arc seconds to radians
      DOUBLE PRECISION DAS2R
      PARAMETER ( DAS2R = 4.848136811095359935899141D-6 )
```

SOFA C constants

The constants used by the C version of SOFA are defined in the header file sofam.h.

```
#ifndef SOFAHDEF
#define SOFAHDEF

/*
**  - - - - - - -
**   s o f a . h
**  - - - - - - -
**
**  Prototype function declarations for SOFA library.
**
**  This file is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  This revision:   2023 April 16
**
**  SOFA release 2023-10-11
**
**  Copyright (C) 2023 IAU SOFA Board.  See notes at end.
*/

#include "math.h"

#ifdef __cplusplus
extern "C" {
#endif

/* Star-independent astrometry parameters */
typedef struct {
   double pmt;        /* PM time interval (SSB, Julian years) */
   double eb[3];      /* SSB to observer (vector, au) */
   double eh[3];      /* Sun to observer (unit vector) */
   double em;         /* distance from Sun to observer (au) */
   double v[3];       /* barycentric observer velocity (vector, c) */
   double bm1;        /* sqrt(1-|v|^2): reciprocal of Lorenz factor */
   double bpn[3][3];  /* bias-precession-nutation matrix */
   double along;      /* longitude + s' + dERA(DUT) (radians) */
   double phi;        /* geodetic latitude (radians) */
   double xpl;        /* polar motion xp wrt local meridian (radians) */
   double ypl;        /* polar motion yp wrt local meridian (radians) */
   double sphi;       /* sine of geodetic latitude */
   double cphi;       /* cosine of geodetic latitude */
   double diurab;     /* magnitude of diurnal aberration vector */
   double eral;       /* "local" Earth rotation angle (radians) */
   double refa;       /* refraction constant A (radians) */
   double refb;       /* refraction constant B (radians) */
} iauASTROM;
/* (Vectors eb, eh, em and v are all with respect to BCRS axes.) */

/* Body parameters for light deflection */
typedef struct {
   double bm;         /* mass of the body (solar masses) */
   double dl;         /* deflection limiter (radians^2/2) */
   double pv[2][3];   /* barycentric PV of the body (au, au/day) */
} iauLDBODY;

/* Astronomy/Calendars */
int iauCal2jd(int iy, int im, int id, double *djm0, double *djm);
double iauEpb(double dj1, double dj2);
void iauEpb2jd(double epb, double *djm0, double *djm);
double iauEpj(double dj1, double dj2);
void iauEpj2jd(double epj, double *djm0, double *djm);
int iauJd2cal(double dj1, double dj2,
                     int *iy, int *im, int *id, double *fd);
int iauJdcalf(int ndp, double dj1, double dj2, int iymdf[4]);

/* Astronomy/Astrometry */
void iauAb(double pnat[3], double v[3], double s, double bm1,
           double ppr[3]);
void iauApcg(double date1, double date2,
             double ebpv[2][3], double ehp[3],
             iauASTROM *astrom);
void iauApcg13(double date1, double date2, iauASTROM *astrom);
void iauApci(double date1, double date2,
```

```
             double ebpv[2][3], double ehp[3],
             double x, double y, double s,
             iauASTROM *astrom);
void iauApci13(double date1, double date2,
               iauASTROM *astrom, double *eo);
void iauApco(double date1, double date2,
             double ebpv[2][3], double ehp[3],
             double x, double y, double s, double theta,
             double elong, double phi, double hm,
             double xp, double yp, double sp,
             double refa, double refb,
             iauASTROM *astrom);
int iauApco13(double utc1, double utc2, double dut1,
              double elong, double phi, double hm, double xp, double yp,
              double phpa, double tc, double rh, double wl,
              iauASTROM *astrom, double *eo);
void iauApcs(double date1, double date2, double pv[2][3],
             double ebpv[2][3], double ehp[3],
             iauASTROM *astrom);
void iauApcs13(double date1, double date2, double pv[2][3],
               iauASTROM *astrom);
void iauAper(double theta, iauASTROM *astrom);
void iauAper13(double ut11, double ut12, iauASTROM *astrom);
void iauApio(double sp, double theta,
             double elong, double phi, double hm, double xp, double yp,
             double refa, double refb,
             iauASTROM *astrom);
int iauApio13(double utc1, double utc2, double dut1,
              double elong, double phi, double hm, double xp, double yp,
              double phpa, double tc, double rh, double wl,
              iauASTROM *astrom);
void iauAtcc13(double rc, double dc,
               double pr, double pd, double px, double rv,
               double date1, double date2,
               double *ra, double *da);
void iauAtccq(double rc, double dc,
              double pr, double pd, double px, double rv,
              iauASTROM *astrom, double *ra, double *da);
void iauAtci13(double rc, double dc,
               double pr, double pd, double px, double rv,
               double date1, double date2,
               double *ri, double *di, double *eo);
void iauAtciq(double rc, double dc, double pr, double pd,
              double px, double rv, iauASTROM *astrom,
              double *ri, double *di);
void iauAtciqn(double rc, double dc, double pr, double pd,
               double px, double rv, iauASTROM *astrom,
               int n, iauLDBODY b[], double *ri, double *di);
void iauAtciqz(double rc, double dc, iauASTROM *astrom,
               double *ri, double *di);
int iauAtco13(double rc, double dc,
              double pr, double pd, double px, double rv,
              double utc1, double utc2, double dut1,
              double elong, double phi, double hm, double xp, double yp,
              double phpa, double tc, double rh, double wl,
              double *aob, double *zob, double *hob,
              double *dob, double *rob, double *eo);
void iauAtic13(double ri, double di,
               double date1, double date2,
               double *rc, double *dc, double *eo);
void iauAticq(double ri, double di, iauASTROM *astrom,
              double *rc, double *dc);
void iauAticqn(double ri, double di, iauASTROM *astrom,
               int n, iauLDBODY b[], double *rc, double *dc);
int iauAtio13(double ri, double di,
              double utc1, double utc2, double dut1,
              double elong, double phi, double hm, double xp, double yp,
              double phpa, double tc, double rh, double wl,
              double *aob, double *zob, double *hob,
              double *dob, double *rob);
void iauAtioq(double ri, double di, iauASTROM *astrom,
              double *aob, double *zob,
              double *hob, double *dob, double *rob);
```

```
int iauAtoc13(const char *type, double ob1, double ob2,
              double utc1, double utc2, double dut1,
              double elong, double phi, double hm, double xp, double yp,
              double phpa, double tc, double rh, double wl,
              double *rc, double *dc);
int iauAtoi13(const char *type, double ob1, double ob2,
              double utc1, double utc2, double dut1,
              double elong, double phi, double hm, double xp, double yp,
              double phpa, double tc, double rh, double wl,
              double *ri, double *di);
void iauAtoiq(const char *type,
              double ob1, double ob2, iauASTROM *astrom,
              double *ri, double *di);
void iauLd(double bm, double p[3], double q[3], double e[3],
           double em, double dlim, double p1[3]);
void iauLdn(int n, iauLDBODY b[], double ob[3], double sc[3],
            double sn[3]);
void iauLdsun(double p[3], double e[3], double em, double p1[3]);
void iauPmpx(double rc, double dc, double pr, double pd,
             double px, double rv, double pmt, double pob[3],
             double pco[3]);
int iauPmsafe(double ra1, double dec1, double pmr1, double pmd1,
              double px1, double rv1,
              double ep1a, double ep1b, double ep2a, double ep2b,
              double *ra2, double *dec2, double *pmr2, double *pmd2,
              double *px2, double *rv2);
void iauPvtob(double elong, double phi, double height, double xp,
              double yp, double sp, double theta, double pv[2][3]);
void iauRefco(double phpa, double tc, double rh, double wl,
              double *refa, double *refb);

/* Astronomy/Ephemerides */
int iauEpv00(double date1, double date2,
             double pvh[2][3], double pvb[2][3]);
void iauMoon98(double date1, double date2, double pv[2][3]);
int iauPlan94(double date1, double date2, int np, double pv[2][3]);

/* Astronomy/FundamentalArgs */
double iauFad03(double t);
double iauFae03(double t);
double iauFaf03(double t);
double iauFaju03(double t);
double iauFal03(double t);
double iauFalp03(double t);
double iauFama03(double t);
double iauFame03(double t);
double iauFane03(double t);
double iauFaom03(double t);
double iauFapa03(double t);
double iauFasa03(double t);
double iauFaur03(double t);
double iauFave03(double t);

/* Astronomy/PrecNutPolar */
void iauBi00(double *dpsibi, double *depsbi, double *dra);
void iauBp00(double date1, double date2,
             double rb[3][3], double rp[3][3], double rbp[3][3]);
void iauBp06(double date1, double date2,
             double rb[3][3], double rp[3][3], double rbp[3][3]);
void iauBpn2xy(double rbpn[3][3], double *x, double *y);
void iauC2i00a(double date1, double date2, double rc2i[3][3]);
void iauC2i00b(double date1, double date2, double rc2i[3][3]);
void iauC2i06a(double date1, double date2, double rc2i[3][3]);
void iauC2ibpn(double date1, double date2, double rbpn[3][3],
               double rc2i[3][3]);
void iauC2ixy(double date1, double date2, double x, double y,
              double rc2i[3][3]);
void iauC2ixys(double x, double y, double s, double rc2i[3][3]);
void iauC2t00a(double tta, double ttb, double uta, double utb,
               double xp, double yp, double rc2t[3][3]);
void iauC2t00b(double tta, double ttb, double uta, double utb,
               double xp, double yp, double rc2t[3][3]);
void iauC2t06a(double tta, double ttb, double uta, double utb,
```

```
               double xp, double yp, double rc2t[3][3]);
void iauC2tcio(double rc2i[3][3], double era, double rpom[3][3],
               double rc2t[3][3]);
void iauC2teqx(double rbpn[3][3], double gst, double rpom[3][3],
               double rc2t[3][3]);
void iauC2tpe(double tta, double ttb, double uta, double utb,
              double dpsi, double deps, double xp, double yp,
              double rc2t[3][3]);
void iauC2txy(double tta, double ttb, double uta, double utb,
              double x, double y, double xp, double yp,
              double rc2t[3][3]);
double iauEo06a(double date1, double date2);
double iauEors(double rnpb[3][3], double s);
void iauFw2m(double gamb, double phib, double psi, double eps,
             double r[3][3]);
void iauFw2xy(double gamb, double phib, double psi, double eps,
              double *x, double *y);
void iauLtp(double epj, double rp[3][3]);
void iauLtpb(double epj, double rpb[3][3]);
void iauLtpecl(double epj, double vec[3]);
void iauLtpequ(double epj, double veq[3]);
void iauNum00a(double date1, double date2, double rmatn[3][3]);
void iauNum00b(double date1, double date2, double rmatn[3][3]);
void iauNum06a(double date1, double date2, double rmatn[3][3]);
void iauNumat(double epsa, double dpsi, double deps, double rmatn[3][3]);
void iauNut00a(double date1, double date2, double *dpsi, double *deps);
void iauNut00b(double date1, double date2, double *dpsi, double *deps);
void iauNut06a(double date1, double date2, double *dpsi, double *deps);
void iauNut80(double date1, double date2, double *dpsi, double *deps);
void iauNutm80(double date1, double date2, double rmatn[3][3]);
double iauObl06(double date1, double date2);
double iauObl80(double date1, double date2);
void iauP06e(double date1, double date2,
             double *eps0, double *psia, double *oma, double *bpa,
             double *bqa, double *pia, double *bpia,
             double *epsa, double *chia, double *za, double *zetaa,
             double *thetaa, double *pa,
             double *gam, double *phi, double *psi);
void iauPb06(double date1, double date2,
             double *bzeta, double *bz, double *btheta);
void iauPfw06(double date1, double date2,
              double *gamb, double *phib, double *psib, double *epsa);
void iauPmat00(double date1, double date2, double rbp[3][3]);
void iauPmat06(double date1, double date2, double rbp[3][3]);
void iauPmat76(double date1, double date2, double rmatp[3][3]);
void iauPn00(double date1, double date2, double dpsi, double deps,
             double *epsa,
             double rb[3][3], double rp[3][3], double rbp[3][3],
             double rn[3][3], double rbpn[3][3]);
void iauPn00a(double date1, double date2,
              double *dpsi, double *deps, double *epsa,
              double rb[3][3], double rp[3][3], double rbp[3][3],
              double rn[3][3], double rbpn[3][3]);
void iauPn00b(double date1, double date2,
              double *dpsi, double *deps, double *epsa,
              double rb[3][3], double rp[3][3], double rbp[3][3],
              double rn[3][3], double rbpn[3][3]);
void iauPn06(double date1, double date2, double dpsi, double deps,
             double *epsa,
             double rb[3][3], double rp[3][3], double rbp[3][3],
             double rn[3][3], double rbpn[3][3]);
void iauPn06a(double date1, double date2,
              double *dpsi, double *deps, double *epsa,
              double rb[3][3], double rp[3][3], double rbp[3][3],
              double rn[3][3], double rbpn[3][3]);
void iauPnm00a(double date1, double date2, double rbpn[3][3]);
void iauPnm00b(double date1, double date2, double rbpn[3][3]);
void iauPnm06a(double date1, double date2, double rnpb[3][3]);
void iauPnm80(double date1, double date2, double rmatpn[3][3]);
void iauPom00(double xp, double yp, double sp, double rpom[3][3]);
void iauPr00(double date1, double date2,
             double *dpsipr, double *depspr);
void iauPrec76(double date01, double date02,
```

```
                double date11, double date12,
                double *zeta, double *z, double *theta);
double iauS00(double date1, double date2, double x, double y);
double iauS00a(double date1, double date2);
double iauS00b(double date1, double date2);
double iauS06(double date1, double date2, double x, double y);
double iauS06a(double date1, double date2);
double iauSp00(double date1, double date2);
void iauXy06(double date1, double date2, double *x, double *y);
void iauXys00a(double date1, double date2,
                double *x, double *y, double *s);
void iauXys00b(double date1, double date2,
                double *x, double *y, double *s);
void iauXys06a(double date1, double date2,
                double *x, double *y, double *s);

/* Astronomy/RotationAndTime */
double iauEe00(double date1, double date2, double epsa, double dpsi);
double iauEe00a(double date1, double date2);
double iauEe00b(double date1, double date2);
double iauEe06a(double date1, double date2);
double iauEect00(double date1, double date2);
double iauEqeq94(double date1, double date2);
double iauEra00(double dj1, double dj2);
double iauGmst00(double uta, double utb, double tta, double ttb);
double iauGmst06(double uta, double utb, double tta, double ttb);
double iauGmst82(double dj1, double dj2);
double iauGst00a(double uta, double utb, double tta, double ttb);
double iauGst00b(double uta, double utb);
double iauGst06(double uta, double utb, double tta, double ttb,
                 double rnpb[3][3]);
double iauGst06a(double uta, double utb, double tta, double ttb);
double iauGst94(double uta, double utb);

/* Astronomy/SpaceMotion */
int iauPvstar(double pv[2][3], double *ra, double *dec,
               double *pmr, double *pmd, double *px, double *rv);
int iauStarpv(double ra, double dec,
               double pmr, double pmd, double px, double rv,
               double pv[2][3]);

/* Astronomy/StarCatalogs */
void iauFk425(double r1950, double d1950,
               double dr1950, double dd1950,
               double p1950, double v1950,
               double *r2000, double *d2000,
               double *dr2000, double *dd2000,
               double *p2000, double *v2000);
void iauFk45z(double r1950, double d1950, double bepoch,
               double *r2000, double *d2000);
void iauFk524(double r2000, double d2000,
               double dr2000, double dd2000,
               double p2000, double v2000,
               double *r1950, double *d1950,
               double *dr1950, double *dd1950,
               double *p1950, double *v1950);
void iauFk52h(double r5, double d5,
               double dr5, double dd5, double px5, double rv5,
               double *rh, double *dh,
               double *drh, double *ddh, double *pxh, double *rvh);
void iauFk54z(double r2000, double d2000, double bepoch,
               double *r1950, double *d1950,
               double *dr1950, double *dd1950);
void iauFk5hip(double r5h[3][3], double s5h[3]);
void iauFk5hz(double r5, double d5, double date1, double date2,
               double *rh, double *dh);
void iauH2fk5(double rh, double dh,
               double drh, double ddh, double pxh, double rvh,
               double *r5, double *d5,
               double *dr5, double *dd5, double *px5, double *rv5);
void iauHfk5z(double rh, double dh, double date1, double date2,
               double *r5, double *d5, double *dr5, double *dd5);
int iauStarpm(double ra1, double dec1,
```

```
              double pmr1, double pmd1, double px1, double rv1,
              double ep1a, double ep1b, double ep2a, double ep2b,
              double *ra2, double *dec2,
              double *pmr2, double *pmd2, double *px2, double *rv2);

/* Astronomy/EclipticCoordinates */
void iauEceq06(double date1, double date2, double dl, double db,
               double *dr, double *dd);
void iauEcm06(double date1, double date2, double rm[3][3]);
void iauEqec06(double date1, double date2, double dr, double dd,
               double *dl, double *db);
void iauLteceq(double epj, double dl, double db, double *dr, double *dd);
void iauLtecm(double epj, double rm[3][3]);
void iauLteqec(double epj, double dr, double dd, double *dl, double *db);

/* Astronomy/GalacticCoordinates */
void iauG2icrs(double dl, double db, double *dr, double *dd);
void iauIcrs2g(double dr, double dd, double *dl, double *db);

/* Astronomy/GeodeticGeocentric */
int iauEform(int n, double *a, double *f);
int iauGc2gd(int n, double xyz[3],
             double *elong, double *phi, double *height);
int iauGc2gde(double a, double f, double xyz[3],
              double *elong, double *phi, double *height);
int iauGd2gc(int n, double elong, double phi, double height,
             double xyz[3]);
int iauGd2gce(double a, double f,
              double elong, double phi, double height, double xyz[3]);

/* Astronomy/Timescales */
int iauD2dtf(const char *scale, int ndp, double d1, double d2,
             int *iy, int *im, int *id, int ihmsf[4]);
int iauDat(int iy, int im, int id, double fd, double *deltat);
double iauDtdb(double date1, double date2,
               double ut, double elong, double u, double v);
int iauDtf2d(const char *scale, int iy, int im, int id,
             int ihr, int imn, double sec, double *d1, double *d2);
int iauTaitt(double tai1, double tai2, double *tt1, double *tt2);
int iauTaiut1(double tai1, double tai2, double dta,
              double *ut11, double *ut12);
int iauTaiutc(double tai1, double tai2, double *utc1, double *utc2);
int iauTcbtdb(double tcb1, double tcb2, double *tdb1, double *tdb2);
int iauTcgtt(double tcg1, double tcg2, double *tt1, double *tt2);
int iauTdbtcb(double tdb1, double tdb2, double *tcb1, double *tcb2);
int iauTdbtt(double tdb1, double tdb2, double dtr,
             double *tt1, double *tt2);
int iauTttai(double tt1, double tt2, double *tai1, double *tai2);
int iauTttcg(double tt1, double tt2, double *tcg1, double *tcg2);
int iauTttdb(double tt1, double tt2, double dtr,
             double *tdb1, double *tdb2);
int iauTtut1(double tt1, double tt2, double dt,
             double *ut11, double *ut12);
int iauUt1tai(double ut11, double ut12, double dta,
              double *tai1, double *tai2);
int iauUt1tt(double ut11, double ut12, double dt,
             double *tt1, double *tt2);
int iauUt1utc(double ut11, double ut12, double dut1,
              double *utc1, double *utc2);
int iauUtctai(double utc1, double utc2, double *tai1, double *tai2);
int iauUtcut1(double utc1, double utc2, double dut1,
              double *ut11, double *ut12);

/* Astronomy/HorizonEquatorial */
void iauAe2hd(double az, double el, double phi,
              double *ha, double *dec);
void iauHd2ae(double ha, double dec, double phi,
              double *az, double *el);
double iauHd2pa(double ha, double dec, double phi);

/* Astronomy/Gnomonic */
int iauTpors(double xi, double eta, double a, double b,
             double *a01, double *b01, double *a02, double *b02);
```

```
int iauTporv(double xi, double eta, double v[3],
             double v01[3], double v02[3]);
void iauTpsts(double xi, double eta, double a0, double b0,
              double *a, double *b);
void iauTpstv(double xi, double eta, double v0[3], double v[3]);
int iauTpxes(double a, double b, double a0, double b0,
             double *xi, double *eta);
int iauTpxev(double v[3], double v0[3], double *xi, double *eta);

/* VectorMatrix/AngleOps */
void iauA2af(int ndp, double angle, char *sign, int idmsf[4]);
void iauA2tf(int ndp, double angle, char *sign, int ihmsf[4]);
int iauAf2a(char s, int ideg, int iamin, double asec, double *rad);
double iauAnp(double a);
double iauAnpm(double a);
void iauD2tf(int ndp, double days, char *sign, int ihmsf[4]);
int iauTf2a(char s, int ihour, int imin, double sec, double *rad);
int iauTf2d(char s, int ihour, int imin, double sec, double *days);

/* VectorMatrix/BuildRotations */
void iauRx(double phi, double r[3][3]);
void iauRy(double theta, double r[3][3]);
void iauRz(double psi, double r[3][3]);

/* VectorMatrix/CopyExtendExtract */
void iauCp(double p[3], double c[3]);
void iauCpv(double pv[2][3], double c[2][3]);
void iauCr(double r[3][3], double c[3][3]);
void iauP2pv(double p[3], double pv[2][3]);
void iauPv2p(double pv[2][3], double p[3]);

/* VectorMatrix/Initialization */
void iauIr(double r[3][3]);
void iauZp(double p[3]);
void iauZpv(double pv[2][3]);
void iauZr(double r[3][3]);

/* VectorMatrix/MatrixOps */
void iauRxr(double a[3][3], double b[3][3], double atb[3][3]);
void iauTr(double r[3][3], double rt[3][3]);

/* VectorMatrix/MatrixVectorProducts */
void iauRxp(double r[3][3], double p[3], double rp[3]);
void iauRxpv(double r[3][3], double pv[2][3], double rpv[2][3]);
void iauTrxp(double r[3][3], double p[3], double trp[3]);
void iauTrxpv(double r[3][3], double pv[2][3], double trpv[2][3]);

/* VectorMatrix/RotationVectors */
void iauRm2v(double r[3][3], double w[3]);
void iauRv2m(double w[3], double r[3][3]);

/* VectorMatrix/SeparationAndAngle */
double iauPap(double a[3], double b[3]);
double iauPas(double al, double ap, double bl, double bp);
double iauSepp(double a[3], double b[3]);
double iauSeps(double al, double ap, double bl, double bp);

/* VectorMatrix/SphericalCartesian */
void iauC2s(double p[3], double *theta, double *phi);
void iauP2s(double p[3], double *theta, double *phi, double *r);
void iauPv2s(double pv[2][3],
             double *theta, double *phi, double *r,
             double *td, double *pd, double *rd);
void iauS2c(double theta, double phi, double c[3]);
void iauS2p(double theta, double phi, double r, double p[3]);
void iauS2pv(double theta, double phi, double r,
             double td, double pd, double rd,
             double pv[2][3]);

/* VectorMatrix/VectorOps */
double iauPdp(double a[3], double b[3]);
double iauPm(double p[3]);
void iauPmp(double a[3], double b[3], double amb[3]);
```

```
void iauPn(double p[3], double *r, double u[3]);
void iauPpp(double a[3], double b[3], double apb[3]);
void iauPpsp(double a[3], double s, double b[3], double apsb[3]);
void iauPvdpv(double a[2][3], double b[2][3], double adb[2]);
void iauPvm(double pv[2][3], double *r, double *s);
void iauPvmpv(double a[2][3], double b[2][3], double amb[2][3]);
void iauPvppv(double a[2][3], double b[2][3], double apb[2][3]);
void iauPvu(double dt, double pv[2][3], double upv[2][3]);
void iauPvup(double dt, double pv[2][3], double p[3]);
void iauPvxpv(double a[2][3], double b[2][3], double axb[2][3]);
void iauPxp(double a[3], double b[3], double axb[3]);
void iauS2xpv(double s1, double s2, double pv[2][3], double spv[2][3]);
void iauSxp(double s, double p[3], double sp[3]);
void iauSxpv(double s, double pv[2][3], double spv[2][3]);

#ifdef __cplusplus
}
#endif

#endif
```

```
#ifndef SOFAMHDEF
#define SOFAMHDEF

/*
**  - - - - - - - -
**   s o f a m . h
**  - - - - - - - -
**
**  Macros used by SOFA library.
**
**  This file is part of the International Astronomical Union's
**  SOFA (Standards of Fundamental Astronomy) software collection.
**
**  Please note that the constants defined below are to be used only in
**  the context of the SOFA software, and have no other official IAU
**  status.  In addition, self consistency is not guaranteed.
**
**  This revision:   2021 February 24
**
**  SOFA release 2023-10-11
**
**  Copyright (C) 2023 IAU SOFA Board.  See notes at end.
*/

/* Pi */
#define DPI (3.141592653589793238462643)

/* 2Pi */
#define D2PI (6.283185307179586476925287)

/* Radians to degrees */
#define DR2D (57.29577951308232087679815)

/* Degrees to radians */
#define DD2R (1.745329251994329576923691e-2)

/* Radians to arcseconds */
#define DR2AS (206264.8062470963551564734)

/* Arcseconds to radians */
#define DAS2R (4.848136811095359935899141e-6)

/* Seconds of time to radians */
#define DS2R (7.272205216643039903848712e-5)

/* Arcseconds in a full circle */
#define TURNAS (1296000.0)

/* Milliarcseconds to radians */
#define DMAS2R (DAS2R / 1e3)

/* Length of tropical year B1900 (days) */
#define DTY (365.242198781)

/* Seconds per day. */
#define DAYSEC (86400.0)

/* Days per Julian year */
#define DJY (365.25)

/* Days per Julian century */
#define DJC (36525.0)

/* Days per Julian millennium */
#define DJM (365250.0)

/* Reference epoch (J2000.0), Julian Date */
#define DJ00 (2451545.0)

/* Julian Date of Modified Julian Date zero */
#define DJM0 (2400000.5)

/* Reference epoch (J2000.0), Modified Julian Date */
```

```
#define DJM00 (51544.5)

/* 1977 Jan 1.0 as MJD */
#define DJM77 (43144.0)

/* TT minus TAI (s) */
#define TTMTAI (32.184)

/* Astronomical unit (m, IAU 2012) */
#define DAU (149597870.7e3)

/* Speed of light (m/s) */
#define CMPS 299792458.0

/* Light time for 1 au (s) */
#define AULT (DAU/CMPS)

/* Speed of light (au per day) */
#define DC (DAYSEC/AULT)

/* L_G = 1 - d(TT)/d(TCG) */
#define ELG (6.969290134e-10)

/* L_B = 1 - d(TDB)/d(TCB), and TDB (s) at TAI 1977/1/1.0 */
#define ELB (1.550519768e-8)
#define TDB0 (-6.55e-5)

/* Schwarzschild radius of the Sun (au) */
/* = 2 * 1.32712440041e20 / (2.99792458e8)^2 / 1.49597870700e11 */
#define SRS 1.97412574336e-8

/* dint(A) - truncate to nearest whole number towards zero (double) */
#define dint(A) ((A)<0.0?ceil(A):floor(A))

/* dnint(A) - round to nearest whole number (double) */
#define dnint(A) (fabs(A)<0.5?0.0\
                                :((A)<0.0?ceil((A)-0.5):floor((A)+0.5)))

/* dsign(A,B) - magnitude of A with sign of B (double) */
#define dsign(A,B) ((B)<0.0?-fabs(A):fabs(A))

/* max(A,B) - larger (most +ve) of two numbers (generic) */
#define gmax(A,B) (((A)>(B))?(A):(B))

/* min(A,B) - smaller (least +ve) of two numbers (generic) */
#define gmin(A,B) (((A)<(B))?(A):(B))

/* Reference ellipsoids */
#define WGS84 1
#define GRS80 2
#define WGS72 3

#endif
```

```
**                  United Kingdom
**
**----------------------------------------------------------------------*/
```

IAU STANDARDS OF FUNDAMENTAL ASTRONOMY BOARD

Current Membership

John Bangert – United States Naval Observatory, retired
Steven Bell – His Majesty's Nautical Almanac Office (HMNAO)
Nicole Capitaine – Paris Observatory
Maria Davis – United States Naval Observatory (IERS)
Mickael Gastineau – Paris Observatory, IMCCE
Catherine Hohenkerk – HMNAO (Chair, retired)
Li Jinling – Shanghai Astronomical Observatory
Zinovy Malkin – Pulkovo Observatory, St Petersburg
Jeffrey Percival – University of Wisconsin
Wendy Puatua – United States Naval Observatory
Scott Ransom – National Radio Astronomy Observatory
Nick Stamatakos – United States Naval Observatory
Patrick Wallace – RAL Space, retired
Toni Wilmot – His Majesty's Nautical Almanac Office

Past Members

Wim Brouw	University of Groningen
Mark Calabretta	Australia Telescope National Facility
William Folkner	Jet Propulsion Laboratory
Anne-Marie Gontier	Paris Observatory
George Hobbs	Australia Telescope National Facility
George Kaplan	United States Naval Observatory
Brian Luzum	United States Naval Observatory
Dennis McCarthy	United States Naval Observatory
Skip Newhall	Jet Propulsion Laboratory
Jin Wen-Jing	Shanghai Astronomical Observatory

The email address for the Board chair is catherine.hohenkerk@gmail.com