International Astronomical Union

Standards Of Fundamental Astronomy

SOFA Time Scale and Calendar Tools

Software version 13
Document revision 1.6
Version for C programming language

http://www.iausofa.org

2020 December 14

MEMBERS OF THE IAU SOFA BOARD (2010)

John Bangert	United States Naval Observatory (retired)
Steven Bell	Her Majesty's Nautical Almanac Office
Mark Calabretta	Australia Telescope National Facility
Nicole Capitaine	Paris Observatory
William Folkner	Jet Propulsion Laboratory
George Hobbs	Australia Telescope National Facility
Catherine Hohenkerk	Her Majesty's Nautical Almanac Office (chair)
Jin Wen-Jing	Shanghai Observatory
Brian Luzum	United States Naval Observatory (IERS)
Zinovy Malkin	Pulkovo Observatory, St Petersburg
Jeffrey Percival	University of Wisconsin
Patrick Wallace	RAL Space (retired)

Past Members

Wim Brouw	University of Groningen
Anne-Marie Gontier	Paris Observatory
George Kaplan	United States Naval Observatory
Dennis McCarthy	United States Naval Observatory
Skip Newhall	Jet Propulsion Laboratory

Contents

1 Preliminaries

1.1 Introduction

SOFA stands for *Standards Of Fundamental Astronomy.* The SOFA software is a collection of Fortran 77 and C subprograms that implement official IAU algorithms for fundamental-astronomy computations. At the present time the SOFA software comprises 189 astronomy functions supported by 55 utility (mainly vector/matrix) functionsThe core documentation for the SOFA collection consists of classified and alphabetic lists of function calls plus detailed preamble comments in the source code of individual functions.

The present document looks at a selection of SOFA functions that deal with dates and times. It provides a tutorial introduction to timekeeping and the principal time scales and calendars used by astronomers. Short examples demonstrate how to call SOFA functions to perform the sorts of conversions and transformations that may be needed in applications where times and dates are involved.

1.2 Quick start

Readers already familiar with the elementary concepts can safely omit the explanatory material and refer directly to the examples in Sections 1.5 and 4.4, and others that can be found embedded in the text. Figure 2 shows the supported time scales and the names of the transformation functions.

1.3 The SOFA time and date functions

The SOFA functions to be discussed are the following:

`iauCal2jd`	Gregorian calendar to Julian day number
`iauD2tf`	decompose days into hours, minutes, seconds
`iauD2dtf`	decode internal format into time and date numbers
`iauDat`	ΔAT (= TAI−UTC) for a given UTC date
`iauDtdb`	TDB−TT
`iauDtf2d`	encode time and date numbers into internal format
`iauEpb`	Julian date to Besselian epoch
`iauEpb2jd`	Besselian epoch to Julian date
`iauEpj`	Julian date to Julian epoch
`iauEpj2jd`	Julian epoch to Julian date
`iauJd2cal`	Julian date to Gregorian year, month, day, fraction
`iauJdcalf`	Julian date to Gregorian date for formatted output
`iauTaitt`	TAI to TT
`iauTaiut1`	TAI to UT1
`iauTaiutc`	TAI to UTC
`iauTcbtdb`	TCB to TDB

`iauTcgtt`	TCG to TT
`iauTdbtcb`	TDB to TCB
`iauTdbtt`	TDB to TT
`iauTf2d`	hours, minutes, seconds into days
`iauTttai`	TT to TAI
`iauTttcg`	TT to TCG
`iauTttdb`	TT to TDB
`iauTtut1`	TT to UT1
`iauUt1tai`	UT1 to TAI
`iauUt1tt`	UT1 to TT
`iauUt1utc`	UT1 to UTC
`iauUtctai`	UTC to TAI
`iauUtcut1`	UTC to UT1

Detailed specifications for all of these are provided in Section 6.

1.4 Intended audience

The SOFA time functions are designed for convenient practical use, with levels of rigor that are consistent with this goal. Inevitably, many fine details are glossed over, such as transforming topocentric proper time to TAI, different realizations of TT, competing TDB−TT models, and so on. Furthermore, the "proleptic" issue is largely ignored; for instance the functions will blithely perform a transformation from TAI to TT for a date in the nineteenth century, long before either time scale was introduced. These simplifications notwithstanding, time specialists may nevertheless find SOFA useful as a reliable source of comparison results.

1.5 A simple example: UTC to TT

A particularly common application is when an instant in one time scale is to be referred to another time scale. This requires three steps:

1. Call the `iauDtf2d` function to transform the date and time into the SOFA internal format.
2. Call the appropriate sequence of transformation functions (see Figure 2).
3. Call the `iauD2dtf` function to prepare the transformed time for output.

For example, to transform 2010 July 24, 11:18:07.318 (UTC) into terrestrial time TT, and report it, rounded to 1 ms precision:

```
   int j, iy, im, id, ihmsf[4];
   double u1, u2, a1, a2, t1, t2;


/* Encode UTC date and time into internal format. */
   j = iauDtf2d ( "UTC", 2010, 7, 24, 11, 18, 7.318, &u1, &u2 );
   if ( j ) return 1;

/* Transform UTC to TAI, then TAI to TT. */
   j = iauUtctai ( u1, u2, &a1, &a2 );
   if ( j ) return 1;
   j = iauTaitt ( a1, a2, &t1, &t2 );
   if ( j ) return 1;

/* Decode and report the TT. */
   j = iauD2dtf ( "tt", 3, t1, t2, &iy, &im, &id, ihmsf );
   if ( j ) return 1;
   printf ( "%4d/%2.2d/%2.2d%3d:%2.2d:%2.2d.%3.3d\n", iy, im, id,
            ihmsf[0], ihmsf[1], ihmsf[2], ihmsf[3] );
```

The output is "`2010/07/24 11:19:13.502`".

1.6 Abbreviations

GPS	Global Positioning System
GR	General relativity
IAU	International Astronomical Union
IERS	International Earth Rotation and reference systems Service
JD	Julian date
J2000.0	2000 January 1.5 (in some specified time scale)
MJD	Modified Julian date
SOFA	Standards of Fundamental Astronomy
TAI	International Atomic Time
TCB	Barycentric Coordinate Time
TCG	Geocentric Coordinate Time
TDB	Barycentric Dynamical Time
TT	Terrestrial Time
UT, UT1	Universal Time
UTC	Coordinated Universal Time

2 Times and dates

2.1 Timekeeping basics

Timekeeping means following an agreed recipe for measuring time, using some natural "clock" as a basis. The most practical phenomena for this purpose are rotations and oscillations, and for most of history the Earth's rotation was the best available timekeeper. Because time was synonymous with the cycle of day and night, the units used to express time intervals reflect that choice: days, divided into hours, minutes and seconds. For larger intervals, other astronomical phenomena play a role, in particular the orbital periods of the Earth and Moon.

Repeated attempts in the 19th century to model the motion of solar system objects, especially the fast-moving Moon, were only partially successful. Each new theory, while reproducing existing data accurately, would soon start to diverge from observation. The pragmatic solution was to create a tautology by regarding the theories as clocks in their own right; in effect the clock's hands were the bodies of the solar system, reading out "ephemeris time". However, once man-made clocks became sufficiently accurate, suspicions that irregularities in Earth rotation had all along been the limiting factor were confirmed, and laboratory time scales created by such clocks—nowadays atomic, before that quartz—took over as the primary timekeeping standard.

2.2 Formatting conventions

A consequence of the history of timekeeping (and, in the case of the calendar, the nature of the problem) is that the rules for expressing and transforming times and dates are somewhat complicated and inconvenient.

To convert a time interval between hours, minutes, seconds and days, SOFA provides two functions: `iauD2tf` breaks an interval into hours, minutes and seconds while `iauTf2d` does the reverse. Here is an example, where a time is expressed as a fraction of a day; a moment $0^{\mathrm{d}}.25$ earlier is then expressed in hours, minutes and seconds, to a resolution of 1 ms:

```
   char pm;
   int ih, im, j, ihmsf[4];
   double s, f;


/* The time. */
   ih = 23;
   im = 5;
   s = 11.630799;
   printf ( "%2d:%2.2d:%9.6f\n", ih, im, s );

/* Express as a fraction of 1 day. */
   j = iauTf2d ( '+', ih, im, s, &f );
   if ( j ) return 1;
```

```
   printf ( "%14.12f\n", f );

/* Six hours earlier. */
   f -= 0.25;

/* Report to 1 ms precision. */
   iauD2tf ( 3, f, &pm, ihmsf );
   printf ( "%2d:%2.2d:%2.2d.%3.3d\n",
            ihmsf[0], ihmsf[1], ihmsf[2], ihmsf[3] );
```

The output (*i.e.* test time, equivalent fraction of a day, time six hours earlier) is:

```
23:05:11.630799
0.961940171285
17:05:11.631
```

The familiar *Gregorian calendar date*, consisting of year, month and day, is designed to keep more or less in step with the *tropical year*, the year of the seasons, which has a length of about 365.2422 days. Its predecessor, the Julian calendar, approximated the tropical year by using a basic 365-day year and introducing an extra day (February 29) in every fourth year, giving an average year length of 365.25 days. Astronomers still use the latter, the terms *Julian year* and *Julian century* meaning exactly 365.25 and 36525 days respectively. The Gregorian calendar provided a better approximation to the tropical year by dropping three such leap years in each 400 years, to give an average of 365.2425 days. The rule is that century years must be divisible by 400 to be a leap year; other years are leap years if they are divisible by 4. So the year 2000 was a leap year, but 2100 will not be a leap year.

2.3 Julian date

For many purposes, calendar date is inconvenient: what is needed is a continuous count of days. For this purpose the system of *Julian day number* can be used. JD zero is located about 7000 years ago, well before the historical era, and is formally defined in terms of Greenwich noon;[1] for example Julian day number 2449444 began at noon on 1994 April 1.

Julian date (JD) is the same system but with a fractional part appended; JD 2449443.5 was the midnight on which 1994 April 1 commenced. Because of the unwieldy size of Julian Dates and the awkwardness of the half-day offset, it is accepted practice to remove the leading '24' and the trailing '.5', producing what is called the *Modified Julian Date*:

$$\mathrm{MJD} = \mathrm{JD} - 2400000.5.$$

[1] Use of JD and MJD is not restricted to "Greenwich time" and can be used in conjunction with other time scales (Section 3.2) such as TAI, TT and TDB. However, UTC poses problems because of the ambiguity that occurs during a leap second (Section 3.5.1).

Thus 1994 April 1 commenced at MJD 49443.0.

MJD is often used in computer applications, rather than JD itself, to reduce exposure to rounding errors. SOFA goes one step further, by always expressing JD as two double precision numbers, the sum of which is the desired JD. The user decides how best to apportion the JD. In applications where precision is not critical, one of the two parts can simply be set to zero and the other to the JD itself. A better compromise might be 2400000.5 as one of the numbers, and the MJD as the other. The various SOFA functions go to some trouble to use the two parts wisely—they are not simply added together—and the preamble comments in the various functions indicate which split will give optimal accuracy.

The SOFA `iauCal2jd` function expresses a given Gregorian calendar date as a two-part JD. The `iauJd2cal` function takes a two-part JD and breaks it into Gregorian calendar date plus a fraction of a day. The `iauJdcalf` function does a similar job, but rounds to a specified precision and returns the fraction as an integer, ready for use in a message or report. Here is a demonstration, using a deliberately awkward date and time close to the end of a month:

```
   int iy, im, id, ihour, imin, j, iymdf[4];
   double d1, d2, sec, d, fd;


/* Date and time. */
   iy = 2008; im = 2; id = 29;
   ihour = 23; imin = 59; sec = 59.9;
   printf ( "%4d/%2.2d/%2.2d%3d:%2.2d:%4.1f\n",
            iy, im, id, ihour, imin, sec );

/* Express as two-part JD. */
   j = iauCal2jd ( iy, im, id, &d1, &d2 );
   if ( j ) return 1;
   j = iauTf2d ( '+', ihour, imin, sec, &d );
   if ( j ) return 1;
   d2 += d;
   printf ( "%9.1f +%13.6f =%15.6f\n", d1, d2, d1 + d2 );

/* Express as calendar date and fraction of a day. */
   j = iauJd2cal ( d1, d2, &iy, &im, &id, &fd );
   if ( j ) return 1;
   d = ( (double) id ) + fd;
   printf ( "%4d/%2.2d/%9.6f\n", iy, im, d );

/* Round to 0.001 day. */
   j = iauJdcalf ( 3, d1, d2, iymdf );
   if ( j ) return 1;
   printf ( "%4d/%2.2d/%2.2d.%3.3d\n",
            iymdf[0], iymdf[1], iymdf[2], iymdf[3] );
```

The output (*i.e.* date and time, MJD and JD equivalents, date with fractional day, the same but rounded) is:

```
2008/02/29 23:59:59.9
2400000.5 + 54525.999999 = 2454526.499999
2008/02/29.999999
2008/03/01.000
```

2.4 Besselian and Julian epochs

For some astronomical purposes it is convenient to work in fractional years, such that a given date and time near the end of 2009 (for example) can be written "2009.93".

Formerly, this was done using a system called *Besselian epoch*. The unit is tropical years (about 365.2422 days), the time scale is *ephemeris time* (Section 3.2) and the Besselian year begins when the ecliptic longitude of the mean Sun is 280°—which occurs near the start of the calendar year and which is why that particular round figure was chosen. Besselian epochs are mainly associated with older star catalogues, specifying the mean equator and equinox and hence the celestial orientation of the equatorial coordinate system used by the catalogue concerned.

Julian epoch took over from the beginning of 1984. It uses the Julian year of exactly 365.25 days, and the TT time scale (Section 3.2); Julian epoch 2000.0 is defined to be 2000 January 1.5, which is JD 2451545.0 or MJD 51544.5.

The two types of epoch are denoted by a prefix 'B' or 'J', hence "B1950.0" and "J2000.0". In the absence of such a prefix, it can be assumed that epochs before 1984.0 are Besselian and, from 1984.0 on, Julian. The transformations between MJD and the two types of epoch are carried out by the SOFA functions `iauEpb`, `iauEpb2jd`, `iauEpj` and `iauEpj2jd`. They are demonstrated in the following code:

```
   double d, e, d1, d2;


/* Julian Date. */
   d = 2457073.05631;
   printf ( "%13.5f\n", d );

/* Transform into Besselian epoch. */
   e = iauEpb ( 0.0, d );
   printf ( "B%15.10f\n", e );

/* Transform back. */
   iauEpb2jd ( e, &d1, &d2 );
   printf ( "%17.9f\n", d1+d2 );
```

```
/* The same for Julian epoch. */
   e = iauEpj ( 0.0, d );
   printf ( "J%15.10f\n", e );
   iauEpj2jd ( e, &d1, &d2 );
   printf ( "%17.9f\n", d1+d2 );
```

The output (*i.e.* given JD, Besselian epoch, equivalent JD, Julian epoch, equivalent JD) is:

```
2457073.05631
B2015.1365941021
2457073.056310000
J2015.1349933196
2457073.056310000
```

---

3 Time scales

3.1 Time in astronomy

Calculations in any scientific discipline may involve precise time, but what sets astronomy apart is the number and variety of *time scales* that have to be used. There are several reasons for this: astronomers must continue to deal with the very phenomena that lie behind obsolete time scales, in particular the rotation of the Earth and the motions of the planets; as new time scales have been introduced, continuity with the past has been preserved, leaving in the various astronomical time scales a fossil record of former offsets and rates; and in astronomical applications the physical context of the "clock" matters, whether it is on Earth, moving or stationary, or on a spacecraft.

3.2 SOFA support for time scales

SOFA provides functions to handle the following seven time scales, all important to astronomers:

- TAI (International Atomic Time): the official timekeeping standard.
- UTC (Coordinated Universal Time): the basis of civil time.
- UT1 (Universal Time): based on Earth rotation.
- TT (Terrestrial Time): used for solar system ephemeris look-up.[2]
- TCG (Geocentric Coordinate Time): used for calculations centered on the Earth in space.
- TCB (Barycentric Coordinate Time): used for calculations beyond Earth orbit.
- TDB (Barycentric Dynamical Time): a scaled form of TCB that keeps in step with TT on the average.

Note that the above list does not include:

- UT0, UT2: specialist forms of universal time that take into account polar motion and known seasonal effects; no longer used.
- Sidereal time—which is an angle rather than a time.[3]
- GMT (Greenwich mean time): an obsolete time scale that can be taken to mean either UTC or UT1.
- ET (ephemeris time): superseded by TT and TDB.

[2] Strictly TDB, but for most applications TT is more than good enough. See Section 3.6.

[3] The same can be said of UT1; however, the interrelation between UTC and UT1 makes it clearer and more convenient to treat the latter as a time in the present document. Sidereal time is covered in the document *SOFA Tools for Earth Attitude*.

- TDT (terrestrial dynamical time): the former name of TT.
- Civil time (BST, PDT *etc.*): these are UTC offset by some number of hours (or half hours, or even quarter hours, in some cases), depending on longitude and time of year.
- Solar (sundial) time.
- GPS time: strictly speaking not itself an officially recognized time scale, but an accurate and inexpensive way of bringing precise time into an observatory (see Section 3.4).

Of the seven time scales to be described here, one is atomic time (TAI), one is solar time (UT1), one is an atomic/solar hybrid (UTC) and four are dynamical times (TT, TCG, TCB, TDB). Each has a distinct role, and there are offsets of tens of seconds between some of them: when planning an astronomical calculation it is vital to choose the right one. A particularly common mistake is to assume that there is just one sort of precise time, namely UTC, compatible with everything from telescope pointing (which actually requires UT1) to looking up planetary positions (which requires TDB, which may be approximated by TT). In fact UTC itself is almost never the time scale to use for astronomical calculations, except perhaps for record-keeping.

Figure 1 is a "route map" showing how the seven SOFA-supported time scales are related, and how a time on one scale can be transformed into the same time on another scale. The time scales are represented by the rectangular boxes, and their relationships are shown in the rounded boxes. *n.b.* Both the diagram and the text gloss over many fine details, in order to concentrate on practical applications.

The transformation from one time scale to the next can take a number of forms. In some cases, for example TAI to TT, it is simply a fixed offset. In others, for example TAI to UT1, it is an offset that depends on observations and cannot be predicted in advance (or only partially). Some time scales, for example TT and TCG, are linearly related, with a rate change as well as an offset. Others, for example TCG and TCB, require a 4-dimensional spacetime transformation.

3.3 Relativistic considerations

Although the majority of astronomical calculations do not explicitly involve general relativity (GR), and can pretend that all points in space share an absolute time, it is wise to be aware of how GR underpins the scheme set out in Figure 1.

In the spacetime coordinates (t, x, y, z) of an event, the temporal coordinate, t, is the *proper* time that would be read by a (perfect) clock at that location. But in some applications, such as the analysis of pulsar observations, it is convenient for the proper time to be remapped into a differently curved spacetime, in which analysis is simpler and physical interpretation more direct: this introduces the so-called *coordinate* time scales TCG, TCB and TDB. Proper time is in fact the exception rather than the rule: even TAI is, strictly speaking, a coordinate time scale, and the proper time read by the observer's clock must in principle undergo a preliminary site-dependent GR transformation before the diagram can be applied. But for terrestrial observers proper time can, in practice, be regarded as the same as TAI, true to picosecond accuracy in the case of clocks at sea level.

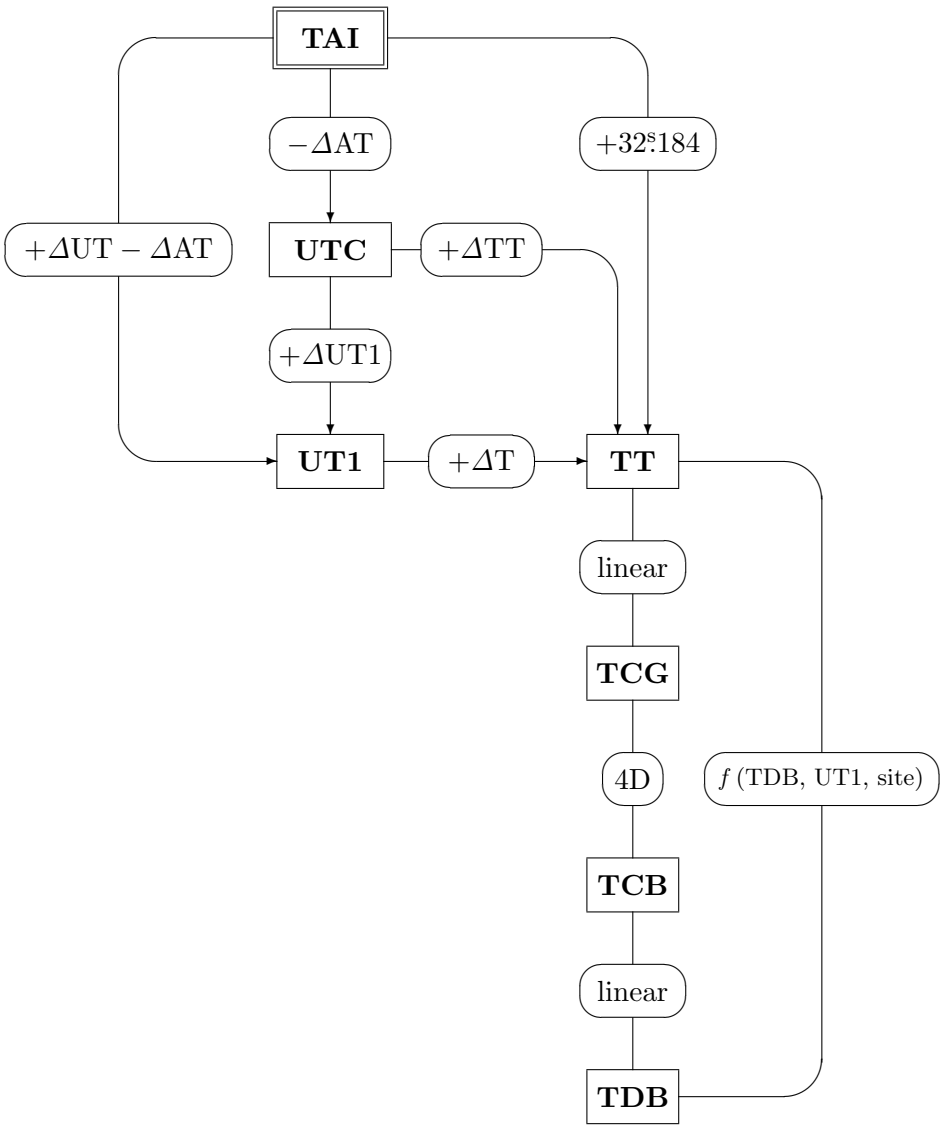


Figure 1: **Relationships between SOFA-supported time scales**

Note that Figure 1 depicts the time of a unique event in a single place (not necessarily on the Earth), expressed in different forms; dubious concepts such as "the time read by a clock at the barycenter" play no part, nor are light-time effects taken into account. For any event, anywhere, it is possible to assign a TAI, a TT, a TDB and so on.

3.4 Atomic time: TAI

The unit of proper time is the SI second, defined as "the duration of 9,192,631,770 periods of the radiation corresponding to the transition between the two hyperfine levels of the ground state of the caesium 133 atom", the latter being at 0 K and in zero magnetic field. The duration was chosen to match the existing astronomical time scale, and is consequently a fossil of the solar day length during the 19th century. The SI second is the unit of TAI and UTC, and is inherited by the relativistic time scales TCG and TCB.[4]

TAI is a laboratory time scale, independent of astronomical phenomena apart from having been synchronized to solar time when first introduced (at the start of 1958). It is realized through a changing population of over 400 high-precision atomic clocks held at standards institutes in various countries. There is an elaborate process of continuous intercomparison, leading to a weighted average of all the clocks involved. TAI is close to proper time for an observer on the geoid, and is an appropriate choice for terrestrial applications where continuity through UTC leap seconds (see 3.5.1) is a requirement. It is not disseminated directly as a time service, but can easily be realized from GPS or UTC.[5] The SOFA software supports transformations between TAI and three other time scales, namely UTC, UT1 and TT.

3.5 Solar Time: UT1 and UTC

UT1 (or plain UT) is the modern equivalent of mean solar time, and is really an angle rather than time in the physics sense. Originally defined in terms of a point in the sky called "the fictitious mean Sun", UT1 is now defined through its relationship with Earth rotation angle (formerly through sidereal time). Because the Earth's rotation rate is slightly irregular—for geophysical reasons—and is gradually decreasing, the UT1 second is not precisely matched to the SI second. This makes UT1 itself unsuitable for use as a time scale in physics applications. However, some applications do require UT1, such as pointing a telescope or antenna at a celestial target, delay calculations in interferometers, and diurnal aberration, parallax and Doppler corrections.

UTC is a compromise between the demands of precise timekeeping and the desire to maintain the current relationship between civil time and daylight. Since its introduction in 1960, UTC has been kept roughly in step with UT1 by a variety of adjustments agreed in advance and

[4] The other two relativistic time scales, TT and TDB, can be regarded either as coordinate times in their own right or as TCG and TCB expressed in different units.

[5] The Global Positioning System (GPS) maintains its own atomic time scale, steered to track UTC without any adjustment for leap seconds implemented after 1980. Because GPS time was set to match UTC in 1980, it inherited the TAI−UTC value from that era. Consequently, GPS time remains forever 19^{s} behind TAI, to sub-microsecond accuracy. The main reasons SOFA does not provide support for GPS time are (i) the transformation, namely a fixed 19^{s} offset, is trivial, and (ii) the danger of confusion between GPS system time itself and the time displayed by most GPS receivers, which typically is UTC.

then carried out in a coordinated manner by the providers of time services—hence the name. Though rate changes were used until 1972, since then all such adjustments have been made by occasionally inserting a whole second, called a *leap second*, a procedure that can be thought of as stopping the UTC clock for a second to let the Earth catch up. Leap seconds are discussed below.

To obtain UT1 starting from UTC, it is necessary to look up the value of $\Delta\mathrm{UT1} = UT1\text{–}UTC$ for the date concerned in tables published by the International Earth Rotation and Reference Systems Service (IERS); this is then added to the UTC. The quantity UT1−UTC, which typically changes by 1-2 ms per day, can be obtained only by observation, principally very long baseline interferometry (VLBI) using extragalactic radio sources, though seasonal effects are present and the IERS listings are able to predict some way into the future with adequate accuracy for most applications.

There are SOFA functions to link UTC, UT1, TAI and TT.

3.5.1 Leap seconds

Note that in Figure 1 there are two ways of getting from TAI to UT1, one via UTC and the other directly. In a practical application (such as a telescope control system), the direct route has the important advantage that the offset varies in a continuous manner, whereas the route via UTC introduces the complication of dealing with leap seconds.

At the time of writing, leap seconds are introduced as necessary to keep UT1−UTC in the range $\pm 0^{\mathrm{s}}.9$. The decision is made by international agreement, and announced several months in advance by the IERS. Leap seconds occur usually at the end of December or June. Because on the average the solar day is now 1-2 ms longer than the nominal 86,400 SI seconds, accumulating to 1^{s} over a period of 18 months to a few years, leap seconds are in practice always positive; however, provision exists for negative leap seconds if needed. Each time a leap second is introduced, the offset $\Delta\mathrm{AT} = TAI - UTC$ changes by exactly 1^{s}. Up-to-date information on TAI−UTC is available from the IERS.

The form of a leap second can be seen from the following description of the end of 2008:

	UTC	ΔUT1	ΔAT	UT1
		s	s	
2008 Dec 31	23 59 58	−0.593	+33	23 59 57.407
	23 59 59	−0.593	+33	23 59 58.407
	23 59 60	−0.593	+33	23 59 59.407
2009 Jan 1	00 00 00	+0.407	+34	00 00 00.407
	00 00 01	+0.407	+34	00 00 01.407

Note that UTC has to be expressed as hours, minutes and seconds (or at least in seconds in a given day) if leap seconds are to be taken into account in the correct manner. In particular, it is inappropriate to express UTC as a Julian Date, because there will be an ambiguity during a

leap second—so that for example 1994 June 30 $23^{\rm h}\,59^{\rm m}\,60\overset{\rm s}{.}0$ and 1994 July 1 $00^{\rm h}\,00^{\rm m}\,00\overset{\rm s}{.}0$ would *both* come out as MJD 49534.00000—and because subtracting two such JDs would not yield the correct interval in cases that contain leap seconds.

Leap seconds pose tricky problems for software writers, and consequently there are concerns that these events put safety-critical systems at risk. The correct solution is for designers to base such systems on TAI or some other glitch-free time scale, not UTC, but this option is often overlooked until it is too late. As the Earth rotation slows, leap seconds will become ever more frequent, increasing the administrative burden and multiplying the danger of disruption. This has led to proposals to relax the $\pm 0\overset{\rm s}{.}9$ requirement for UT1−UTC, putting off further adjustments for several hundred years, at which point they would amount to timezone changes. The main danger in ceasing leap seconds is that there are in existence unknown numbers of applications that rely on UTC being a reasonable approximation to UT1, and these will gradually fail as the discrepancy grows. However, because it is very likely that leap seconds will soon cease, it is essential that all new applications, however modest their accuracy needs, take UT1−UTC properly into account.

SOFA support for leap seconds is built into the various transformations that link UTC with TAI and UT1. These call a SOFA function that contains the historical list of leap seconds, namely `iauDat` (see Section 4.5.)

3.5.2 Delta T

The difference between UT1 and TT (formerly ET) is called ΔT, and in the present era can be written out as

$$\Delta{\rm T} = TT - UT1 = 32\overset{\rm s}{.}184 + \Delta{\rm AT} - \Delta{\rm UT1}.$$

ΔT is important for interpreting historical observations of solar system phenomena, where modern ephemerides provide the time of the event accurately but its appearance at a specified geographical location depends on knowing the Earth orientation.

Because the Earth's rotation is slowing due to tidal friction, and the rotation rate decreases approximately linearly with time, ΔT increases quadratically. Empirical models for ΔT exist, based on historical records such as eclipse sightings, but there is at present no SOFA support in this area. However, SOFA functions are provided to transform between UT1 and TT given a user-supplied ΔT value.

3.6 The dynamical time scales: TT, TCG, TCB and TDB

TT is a successor to the former Ephemeris Time, and is what is in practice[6] used to look up solar system ephemerides. The coordinate time scales TCG, TCB and TDB are the independent variable in GR-based theories which describe the motions of bodies in the vicinity of the Earth

[6]TDB or a specific time scale belonging to the ephemeris concerned (denoted $\rm T_{eph}$) should really be used. However, going to the trouble of computing TDB or $\rm T_{eph}$ is usually not justified: even for the Moon, the error from using TT instead of TDB is less than 1 milliarcsecond.

(TT, TCG) and in the solar system (TCB, TDB). Of the four, most observational astronomers will need only TT, and perhaps TDB.

TT and TDB are close to each other (less than 2 ms) and run at the same rate as TAI (exactly in the case of TT; on the average in the case of TDB): they are the modern equivalents of ET. TCG and TCB, used in theoretical work, run at different rates and so have long term drifts relative to TAI. The four time scales are tied to TAI though the spacetime event 1997 January 1st $0^{\rm h}$ TAI at the geocenter. The times of that event are as follows:

TAI	1997 Jan 1 00:00:00
TCG	1997 Jan 1 00:00:00 + 32.184 s
TCB	1997 Jan 1 00:00:00 + 32.184 s
TDB	1997 Jan 1 00:00:00 + 32.184 s − 65.5 μs

3.6.1 Using solar system ephemerides: a warning

TT (or more correctly TDB) is the time scale to use when interrogating an ephemeris such as JPL DE405, or when using published formulae to predict the position of the Earth or another solar system body. **It is an extremely common mistake to use UTC**, in which case the results will (at the time of writing) be well over a minute out. (The other common blunder is to use the observer's time rather than when the light left the source.)

3.6.2 TT

Terrestrial time, TT (called TDT between 1984 and 2000), is the theoretical time scale for clocks at sea-level: for practical purposes it is tied to TAI through the fixed formula:[7]

$$\mathrm{TT} = \mathrm{TAI} + 32.184\,\mathrm{s}.$$

Calculating TT from UTC, rather than TAI, requires leap seconds to be taken into account: the quantity $\Delta\mathrm{TT} = \mathrm{TT}-\mathrm{UTC}$ for the UTC in question can be deduced from information provided by the IERS. (See also the example in Section 1.5.)

SOFA functions are provided linking TT with TAI, TCG, TDB and UT1.

3.6.3 GCRS and BCRS, and spacetime units

TCG and TCB are the time coordinates of two IAU spacetime metrics called, respectively, the geocentric and barycentric celestial reference systems (GCRS and BCRS). Dynamics problems

[7] The fixed $32^{\rm s}.184$ offset reflects the history of trying to tie together clock timekeeping with Earth rotation and lunar ephemerides. It retains continuity with the former ET, and represents the drift in mean solar time between the ephemerides constructed in the 19th century and the inception of atomic time in the 1950s.

evaluated in these reference systems use physical constants unchanged from their familiar laboratory values. When TT or TDB are to be used for the time coordinate, it is possible to apply the appropriate scaling factor to the entire metric tensor, to preserve the standard relationship $c = 299\,792\,458\,\mathrm{m\,s^{-1}}$. Under these circumstances, physical values such as mass parameters have to be scaled as well. To avoid confusion over units, it is best to embrace the view that the original proper units, such as SI seconds and metres, still apply, projected into the new metric, and never to attach adjectives to units themselves. For example, we might say "The TDB interval is 21.552044 s." or "The TDB-compatible value for GM_E is $3.98004356 \times 10^{14}\,\mathrm{m^3 s^{-2}}$.".

3.6.4 TCG and TCB

Geocentric coordinate time, TCG, is appropriate for theoretical studies of geocentric ephemerides. Its relationship with TT is this conventional linear transformation:

$$\mathrm{TCG} = \mathrm{TT} + \mathrm{L_G} \times (\mathrm{JD_{TT}} - \mathrm{TT_0}),$$

where $\mathrm{TT_0} = 2443144.5003725$ (*i.e.* TT at 1977 January 1.0 TAI), $\mathrm{JD_{TT}}$ is TT expressed as Julian date, and $\mathrm{L_G} = 6.969290134 \times 10^{-10}$. The rate change $\mathrm{L_G}$ means that TCG gains about $2^{\mathrm{s}}.2$ per century with respect to TT or TAI; this represents the combined effect on the terrestrial clock of the gravitational potential from the Earth and the observatory's diurnal speed.

Barycentric coordinate time, TCB, is appropriate for applications where the observer is imagined to be stationary in the solar system but with all the matter (in particular the Sun) absent. The transformation from TCG to TCB thus takes account of the orbital speed of the geocenter and the gravitational potential from the Sun and planets. The difference between TCG and TCB involves a full 4-dimensional GR transformation:

$$\mathrm{TCB} - \mathrm{TCG} = c^{-2} \left\{ \int_{t_0}^{t} [\frac{v_e^2}{2} + U_{ext}(\mathbf{x}_e)] dt + \mathbf{v}_e \cdot (\mathbf{x} - \mathbf{x}_e) \right\} + O(c^{-4}),$$

where the vectors $\mathbf{x}_e$ and $\mathbf{v}_e$ denote the barycentric position and velocity of the geocenter, the vector $\mathbf{x}$ is the barycentric position of the observer and U_{ext} is the Newtonian potential of all of the solar system bodies apart from the Earth, evaluated at the geocenter. In this formula, t is TCB and t_0 is chosen to be consistent with 1977 January 1.0 TAI. The neglected terms, $O(c^{-4})$, are of order 10^{-16} in rate for terrestrial observers. Note that

(i) $U_{ext}(\mathbf{x}_e, \mathbf{x}_e)$ and $\mathbf{v}_e$ are all ephemeris-dependent, and so the resulting TCB belongs to that particular ephemeris, and

(ii) the term $\mathbf{v}_e \cdot (\mathbf{x} - \mathbf{x}_e)$ is zero at the geocenter.

The linear relationship between TT and TCG is implemented in the SOFA functions `iauTttcg` and `iauTcgtt`. SOFA does not provide direct transformations between TT and TCB, instead working via TDB (see the next Section).

3.6.5 TDB

The combined effect of the terrestrial observer's orbital speed and the gravitational potential from the Sun and planets is that a TCB clock gains on TT or TAI by nearly $0^{s}.5$ per year. Barycentric dynamical time, TDB, suppresses this drift by applying a conventional linear transformation:

$$\mathrm{TDB} = \mathrm{TCB} - \mathrm{L_B} \times (\mathrm{JD_{TCB}} - \mathrm{T_0}) \times 86400 + \mathrm{TDB_0},$$

where $\mathrm{JD_{TCB}}$ is TCB is expressed as Julian date, $\mathrm{T_0} = 2443144.5003725$, $\mathrm{L_B} = 1.550519768 \times 10^{-8}$ and $\mathrm{TDB_0} = -6.55 \times 10^{-5}$ s. TDB thus functions as a coordinate time for barycentric applications but, unlike TCB, stays close to TT on the average. The difference TDB−TT is quasi-periodic, dominated by an annual term of amplitude 1.7 ms. For many purposes, the difference is negligible and TT itself can be used instead of TDB. But for some applications—an example being the long-term analysis of pulse arrival times from millisecond pulsars—the difference cannot be neglected and use of TDB (or TCB) is essential.

When, as is usually the case, TT is available but TDB required, the full transformation chain TT → TCG → TCB → TDB can be compressed into a single step. The resulting adjustment is a quasi-periodic quantity[8] dominated by an annual sinusoid of amplitude 1.7 ms.

The linear relationship between TT and TDB is implemented in the SOFA functions `iauTttdb` and `iauTdbtt`. These require a user-supplied TDB−TT value, and the user is free to choose whichever model or tabulation suits the application in hand. The SOFA function `iauDtdb` provides one such model, and is accurate to a few nanoseconds; see Section 4.3.4 for more information.

[8] The 1979 IAU resolution that introduced TDB stipulated that it differ from TDT by periodic terms only. Although this describes the character of the TDB−TT quantity well, it proved impossible to pin down the statement mathematically, and in 2006 the IAU simplified the definition of TDB by making it an *ad hoc* linear transformation of TCB.

4 The SOFA time scale transformation functions

4.1 Architecture

SOFA provides functions that link adjacent pairs of time scales in Figure 1, leaving it to the user to select those needed to construct the desired chain. This raises the obvious question:

> *Why isn't there an all-purpose SOFA function that simply accepts a time with respect to one nominated scale and returns the time with respect to another nominated scale?*

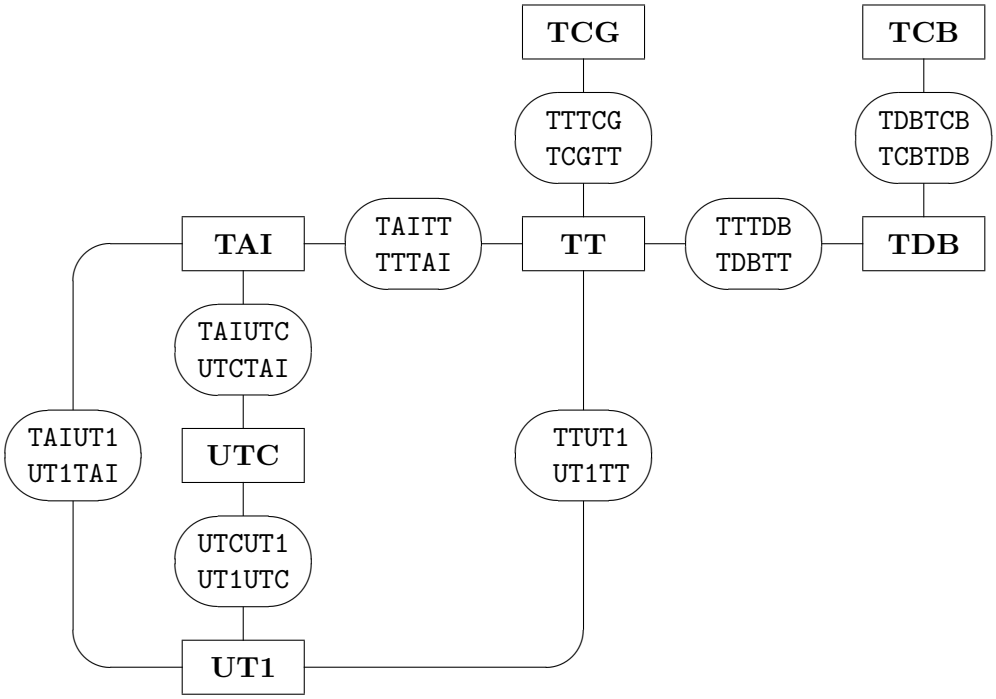
The answer is that some of the transformations involve supplementary quantities that can be provided only by the application, either because they cannot be predicted or because they involve choice. A general-purpose function would need all these quantities as arguments, even though for any given transformation (TAI to TT say) some or all would never be accessed. The example in Section 4.4 gives an idea of what is involved: much of the clutter comes from the TT-to-TDB transformation, which is a function of the observer's position and involves a complicated and computationally heavy ephemeris series. Although the user could simply supply all the quantities every time, just in case, this would be a nuisance, and also computationally wasteful.

Of course, a knowledgeable user would know in advance which quantities are needed and could supply harmless values (zeroes say) for all the others; but a user with that degree of expertise can just as easily build the required chain—and may in any case like to have the intermediate results available. Choosing which functions to call is made easy by Figure 2 which, like Figure 1, shows the transformation chains, but also gives the names of the 16 functions and lists their arguments. An application needing to transform UTC into TDB, for example, could first call `iauUtctai`, then `iauTaitt`, then `iauTttdb`. Only the final call would involve an additional quantity, in this case TDB−TT, which the user could obtain in a number of ways, most conveniently by calling `iauDtdb`.

4.2 Internal representation of times

SOFA has an existing convention for expressing times conveniently and to high precision, namely two-part Julian date (see Section 2.3). The time transformation functions use this convention, so that (for example) the TAI accepted by the `iauTaitt` function is merely an ordinary SOFA two-part JD. However, for reasons already described (Section 3.5.1), it is inappropriate to express UTC as JD; if the straightforward `iauCal2jd`, `iauTf2d`, `iauJd2cal` and `iauJdcalf` functions are used to transform between UTC and JD, there will be an ambiguity during a leap second, the leap second and the subsequent second both returning the same range of JDs.

Rather than using a completely different representation—either for UTC as a special case or for all the transformation functions—the SOFA designers decided instead in the case of UTC to adapt the familiar two-part JD into a form that disguises the leap-second difficulties. This



name	*transformation*	*arguments*
TAIUTC	TAI ⇒ UTC	TAI1, TAI2, UTC1, UTC2
UTCTAI	UTC ⇒ TAI	UTC1, UTC2, TAI1, TAI2
UTCUT1	UTC ⇒ UT1	UTC1, UTC2, DUT, UT1, UT2
UT1UTC	UT1 ⇒ UTC	UT1, UT2, DUT, UTC1, UTC2
TAIUT1	TAI ⇒ UT1	TAI1, TAI2, DTA, UT1, UT2
UT1TAI	UT1 ⇒ TAI	UT1, UT2, DTA, TAI1, TAI2
TTUT1	TT ⇒ UT1	TT1, TT2, DT, UT1, UT2
UT1TT	UT1 ⇒ TT	UT1, UT2, DT, TT1, TT2
TAITT	TAI ⇒ TT	TAI1, TAI2, TT1, TT2
TTTAI	TT ⇒ TAI	TT1, TT2, TAI1, TAI2
TTTCG	TT ⇒ TCG	TT1, TT2, TCG1, TCG2
TCGTT	TCG ⇒ TT	TCG1, TCG2, TT1, TT2
TTTDB	TT ⇒ TDB	TT1, TT2, DTR, TDB1, TDB2
TDBTT	TDB ⇒ TT	TDB1, TDB2, DTR, TT1, TT2
TDBTCB	TDB ⇒ TCB	TDB1, TDB2, TCB1, TCB2
TCBTDB	TCB ⇒ TDB	TCB1, TCB2, TDB1, TDB2

Figure 2: **The SOFA time scale transformation functions.** Argument pairs `TAI1,TAI2` *etc.* are the encoded times produced by the `iauDTF2D` function. `DUT` is UT1−UTC, `DTA` is UT1−TAI, `DT` is TT−UT1, and `DTR` is TDB−TT, all in seconds. (*n.b.* "`TAIUTC`" means C function `iauTaiutc` *etc.)*

modification gives the various time transformation functions a consistent appearance, but at the expense of introducing a non-standard internal scheme in the UTC case.[9]

Two dedicated functions, `iauDtf2d` and `iauD2dtf`, are available to handle the entire encode/decode between formatted date and time—years, month, day, hour, minute and seconds—and two-part JD (or, in the case of UTC, quasi-JD). The identifier for the time scale in question is one of the arguments and, if it is `"UTC"`, special treatment to deal with leap seconds is performed. It is recommended that these functions are used for all date and time encode/decode operations, and mandatory for UTC.

Here is example code that takes a time expressed as TAI, encodes it into the internal format using the `iauDtf2d` function, and transforms it into UTC; the TAI and UTC are each decoded using the `iauD2dtf` function and reported:

```
   int j, iy, im, id, ihmsf[4];
   double a1, a2, u1, u2;


/* Encode TAI date and time into internal format. */
   j = iauDtf2d ( "TAI", 2009, 1, 1, 0, 0, 33.7, &a1, &a2 );
   if ( j ) return 1;

/* Decode and report the TAI. */
   j = iauD2dtf ( "TAI", 3, a1, a2, &iy, &im, &id, ihmsf );
   if ( j ) return 1;
   printf ( "TAI %4d/%2.2d/%2.2d%3d:%2.2d:%2.2d.%3.3d\n",
            iy, im, id, ihmsf[0], ihmsf[1], ihmsf[2], ihmsf[3] );

/* Transform TAI to UTC. */
   j = iauTaiutc ( a1, a2, &u1, &u2 );
   if ( j ) return 1;

/* Decode and report the UTC. */
   j = iauD2dtf ( "UTC", 3, u1, u2, &iy, &im, &id, ihmsf );
   if ( j ) return 1;
   printf ( "UTC %4d/%2.2d/%2.2d%3d:%2.2d:%2.2d.%3.3d\n",
            iy, im, id, ihmsf[0], ihmsf[1], ihmsf[2], ihmsf[3] );
```

The output is:

```
TAI 2009/01/01  0:00:33.700
UTC 2008/12/31 23:59:60.700
```

[9] The quasi-JD that SOFA uses internally to represent UTC is obtained by treating days that end in a leap second as having a different length from usual, namely 86401^{s} (or 86399^{s} in the case of a negative leap second), when forming the fractional part. The scheme has no official status outside SOFA, and it is strongly recommended that applications do **not** exploit it.

Note that the test time is partway through a leap second.

4.3 The supplementary quantities

In Figure 2, while some of the transformations require only a time and the identities of the "before" and "after" time scales, others require additional information that for one reason or another can be supplied only by the application. In the latter cases, the required supplementary quantity has either to be obtained from a service that monitors Earth rotation or involves a choice of model.

4.3.1 UT1 minus UTC

In Figure 2, the argument `DUT` is the quantity $\Delta\mathrm{UT1} = \mathrm{UT1} - \mathrm{UTC}$, in seconds. It can be obtained from tables published by the IERS. For example, the following extract from one of the IERS long-term tabulations covers the date used in the example in Section 4.4, later.

```
                    INTERNATIONAL EARTH ROTATION AND REFERENCE SYSTEMS SERVICE
                         EARTH ORIENTATION PARAMETERS
                           EOP (IERS) 05 C04


              FORMAT(3(I4),I7,2(F11.6),2(F12.7),2(F11.6),2(F11.6),2(F11.7),2F12.6)
******************************************************************************

      Date       MJD       x          y        UT1-UTC       LOD     ...
                           "          "           s           s      ...
     (0h UTC)

1962   1   1  37665  -0.012700   0.213000   0.0326338   0.0017230 ...
1962   1   2  37666  -0.015900   0.214100   0.0320547   0.0016690 ...
1962   1   3  37667  -0.019000   0.215200   0.0315526   0.0015820 ...
  :
2006   1  13  53748   0.048417   0.380702   0.3337689  -0.0002682 ...
2006   1  14  53749   0.048576   0.380378   0.3339950  -0.0001443 ...
2006   1  15  53750   0.049116   0.380121   0.3340826  -0.0000172 ...
2006   1  16  53751   0.049864   0.380044   0.3340667   0.0000669 ...
2006   1  17  53752   0.050329   0.380310   0.3339361   0.0002053 ...
  :
```

4.3.2 UT1 minus TAI

In Figure 2, the argument `DTA` is the quantity UT1−TAI, in seconds. It can be obtained by calling the SOFA function `iauDat` (see Section 4.5) to obtain ΔAT, getting ΔUT1 from IERS tables, and subtracting:

$$\mathrm{UT1} - \mathrm{TAI} = \Delta\mathrm{UT1} - \Delta\mathrm{AT}.$$

4.3.3 TT minus UT1

In Figure 2, the argument `DT` is the quantity $\Delta\mathrm{T} = \mathrm{TT} - \mathrm{UT1}$, in seconds. For current dates it can be constructed accurately by knowing ΔUT1 and ΔAT and using:

$$\Delta\mathrm{T} = 32\overset{\mathrm{s}}{.}184 + \Delta\mathrm{AT} - \Delta\mathrm{UT1}.$$

However, ΔT is more commonly used in applications dealing with observations made many decades or centuries ago, and a suitable value must be chosen by the user, either from tables, for example in *The Astronomical Almanac*, or using a model. (See also Section 3.5.2.)

4.3.4 TDB minus TT

In Figure 2, the argument `DTR` is the quantity TDB−TT, in seconds. The quantity depends on the adopted solar system ephemeris, and is normally obtained either by evaluating a series or by interrogating a precomputed time ephemeris. (See also Section 3.6.5.)

The SOFA function `iauDtdb` provides a detailed model (nearly 800 terms) for TDB−TT, accurate to a few nanoseconds in the present era when compared with the latest work. Its arguments are TDB (TT can be used instead without materially affecting the results), UT1, and the observer's coordinates, expressed as longitude and the distances from the rotation axis and equator. If the application can tolerate errors of up to 2 μs, zeroes can be used for the final four arguments, giving a geocentric result. The example in the next Section demonstrates use of the DTDB routine, including what is involved in transforming the observer's terrestrial coordinates into the required form.

For applications where the distinction between TT and TDB matters but 50 μs accuracy is sufficient, and the date lies in the range 1980-2100, the following simplified expression can be used:

$$\mathrm{TDB} \simeq \mathrm{TT} + 0\overset{\mathrm{s}}{.}001657 \sin g$$

where $g = 6.24 + 0.017202 \times (\mathrm{JD_{TT}} - 2451545)$ approximates the Earth's mean anomaly in radians.

4.4 A comprehensive example: transform UTC into other times

In the following example, an observer at north latitude $+19^\circ\,28'\,52\overset{''}{.}5$, west longitude $155^\circ\,55'\,59\overset{''}{.}6$, at sea level, on 2006 January 15 at 21:24:37.5 UTC, reports the time in all the other supported time scales:

```
   int latnd, latnm, lonwd, lonwm, j, iy, mo, id, ih, im, ihmsf[4];
   double slatn, slonw, hm, elon, phi, xyz[2], u, v, sec,
          utc1, utc2, dut, ut11, ut12, ut, tai1, tai2, tt1, tt2,
          tcg1, tcg2, dtr, tdb1, tdb2, tcb1, tcb2;


/* Site terrestrial coordinates (WGS84). */
   latnd = 19;
   latnm = 28;
   slatn = 52.5;
   lonwd = 155;
   lonwm = 55;
   slonw = 59.6;
   hm = 0.0;

/* Transform to geocentric. */
   j = iauAf2a ( '+', latnd, latnm, slatn, &phi );
   if ( j ) return 1;
   j = iauAf2a ( '-', lonwd, lonwm, slonw, &elon );
   if ( j ) return 1;
   j = iauGd2gc ( 1, elon, phi, hm, xyz );
   if ( j ) return 1;
   u = sqrt ( xyz[0]*xyz[0] + xyz[1]*xyz[1] );
   v = xyz[2];

/* UTC date and time. */
   iy = 2006;
   mo = 1;
   id = 15;
   ih = 21;
   im = 24;
   sec = 37.5;

/* Transform into internal format. */
   j = iauDtf2d ( "UTC", iy, mo, id, ih, im, sec, &utc1, &utc2 );
   if ( j ) return 1;

/* UT1-UTC (s, from IERS). */
   dut = 0.3341;

/* UTC -> UT1. */
   j = iauUtcut1 ( utc1, utc2, dut, &ut11, &ut12 );
   if ( j ) return 1;

/* Extract fraction for TDB-TT calculation, later. */
   ut = fmod ( fmod(ut11,1.0) + fmod(ut12,1.0), 1.0 ) + 0.5;
```

```
/* UTC -> TAI -> TT -> TCG. */
   j = iauUtctai ( utc1, utc2, &tai1, &tai2 );
   if ( j ) return 1;
   j = iauTaitt ( tai1, tai2, &tt1, &tt2 );
   if ( j ) return 1;
   j = iauTttcg ( tt1, tt2, &tcg1, &tcg2 );
   if ( j ) return 1;

/* TDB-TT (using TT as a substitute for TDB). */
   dtr = iauDtdb ( tt1, tt2, ut, elon, u/1e3, v/1e3 );

/* TT -> TDB -> TCB. */
   j = iauTttdb ( tt1, tt2, dtr, &tdb1, &tdb2 );
   if ( j ) return 1;
   j = iauTdbtcb ( tdb1, tdb2, &tcb1, &tcb2 );
   if ( j ) return 1;

/* Report. */
   j = iauD2dtf ( "UTC", 6, utc1, utc2, &iy, &mo, &id, ihmsf );
   if ( j ) return 1;
   printf ( "UTC%5d/%2.2d/%2.2d%3d:%2.2d:%2.2d.%6.6d\n",
            iy, mo, id , ihmsf[0], ihmsf[1], ihmsf[2], ihmsf[3] );

   j = iauD2dtf ( "ut1", 6, ut11, ut12, &iy, &mo, &id, ihmsf );
   if ( j ) return 1;
   printf ( "UT1%5d/%2.2d/%2.2d%3d:%2.2d:%2.2d.%6.6d\n",
            iy, mo, id , ihmsf[0], ihmsf[1], ihmsf[2], ihmsf[3] );

   j = iauD2dtf ( "tai", 6, tai1, tai2, &iy, &mo, &id, ihmsf );
   if ( j ) return 1;
   printf ( "TAI%5d/%2.2d/%2.2d%3d:%2.2d:%2.2d.%6.6d\n",
            iy, mo, id , ihmsf[0], ihmsf[1], ihmsf[2], ihmsf[3] );

   j = iauD2dtf ( "tt", 6, tt1, tt2, &iy, &mo, &id, ihmsf );
   if ( j ) return 1;
   printf ( "TT %5d/%2.2d/%2.2d%3d:%2.2d:%2.2d.%6.6d\n",
            iy, mo, id , ihmsf[0], ihmsf[1], ihmsf[2], ihmsf[3] );

   j = iauD2dtf ( "tcg", 6, tcg1, tcg2, &iy, &mo, &id, ihmsf );
   if ( j ) return 1;
   printf ( "TCG%5d/%2.2d/%2.2d%3d:%2.2d:%2.2d.%6.6d\n",
            iy, mo, id , ihmsf[0], ihmsf[1], ihmsf[2], ihmsf[3] );

   j = iauD2dtf ( "tdb", 6, tdb1, tdb2, &iy, &mo, &id, ihmsf );
   if ( j ) return 1;
   printf ( "TDB%5d/%2.2d/%2.2d%3d:%2.2d:%2.2d.%6.6d\n",
            iy, mo, id , ihmsf[0], ihmsf[1], ihmsf[2], ihmsf[3] );
```

```
   j = iauD2dtf ( "tcb", 6, tcb1, tcb2, &iy, &mo, &id, ihmsf );
   if ( j ) return 1;
   printf ( "TCB%5d/%2.2d/%2.2d%3d:%2.2d:%2.2d.%6.6d\n",
            iy, mo, id , ihmsf[0], ihmsf[1], ihmsf[2], ihmsf[3] );
```

The output is:

```
UTC 2006/01/15 21:24:37.500000
UT1 2006/01/15 21:24:37.834100
TAI 2006/01/15 21:25:10.500000
TT  2006/01/15 21:25:42.684000
TCG 2006/01/15 21:25:43.322690
TDB 2006/01/15 21:25:42.684373
TCB 2006/01/15 21:25:56.893952
```

The UT1 might be used for pointing a telescope, the TT for looking up a planet's position in an ephemeris, and the TDB for interpreting pulsar observations. TAI would be appropriate for calculating long intervals between observed events, also the best way of making time-critical applications leap-second-proof. TCG and TCB would have more specialized dynamical uses. The UTC itself would be suitable for logging and for conversion into local time, and essentially nothing else.

4.5 The iauDat function

Applications that require explicit access to leap second information can call the `iauDat` function. This contains a full list of leap seconds since their inception in 1972, as well as the rate changes used before that. Here is how to obtain TAI−UTC for a given date:

```
   int j;
   double deltat;


/* TAI-UTC for 0h UTC on 2009 Feb 13. */
   j = iauDat ( 2009, 2, 13, 0.0, &deltat );
   if ( j ) return 1;
   printf ( "%+5.1f", deltat );
```

The output is "+34.0".

Note that because the `iauDat` function has to be updated each time a new leap second is announced, there will inevitably be out-of-date copies in circulation, both in source form and inside object libraries. Applications that call `iauDAT`, or that call SOFA functions that themselves call `DAT`, must therefore be relinked occasionally, using object libraries known to have been rebuilt from up-to-date SOFA source code. Rudimentary checks inside the `iauDat` function do at least protect against too large a gap between the nominated date and when the `iauDat` source code was last changed.

However, a better solution for operational systems is to develop a local file or network based scheme, removing any dependence on the SOFA-supplied `iauDat`. To facilitate this, in the special case of the `iauDat` function the standard SOFA licensing rules are relaxed: users are entitled to implement their own `iauDat` function as part of an application that complies with SOFA rules. When this approach is taken, the SOFA-supplied version can still play a useful role as a source of reliable test data.

5 Further reading

Additional explanatory material can be found in *The Astronomical Almanac* and similar national publications. An extensive glossary is available as part of *The Astronomical Almanac Online:* see `http://asa.hmnao.com/SecM/Glossary.html`.

Detailed information on models and procedures can be found in:

- *IERS Conventions (2003),* D. D. McCarthy & G. Petit (eds.), Verlag des Bundesamts für Kartographie und Geodäsie, Frankfurt am Main (2004).

and in successive updates. The following may also be useful for background reading:

- *Explanatory Supplement to the Astronomical Almanac*, ed. Sean E. Urban & P. Kenneth Seidelmann, 3rd Edition (2013), University Science Books.

The previous edition:

- *Explanatory Supplement to the Astronomical Almanac*, ed. P. Kenneth Seidelmann, 2nd Edition (1992), University Science Books.

also contains (Section 2.553) a discussion of ΔT formulas, with reference to the Stephenson & Morrison and McCarthy & Babcock papers mentioned earlier.

6 Function specifications

The following pages present in alphabetical order details of all the SOFA functions that deal with dates and times. Also included are a few other functions that happen to be used in the code examples.

---

iauAf2a *deg, arcmin, arcsec to radians* iauAf2a

CALL :

```
j = iauAf2a ( s, ideg, iamin, asec, &rad );
```

ACTION :

Convert degrees, arcminutes, arcseconds to radians.

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. If the `s` argument is a string, only the leftmost character is used and no warning status is provided.
2. The result is computed even if any of the range checks fail.
3. Negative `ideg`, `iamin` and/or `asec` produce a warning status, but the absolute value is used in the conversion.
4. If there are multiple errors, the status value reflects only the first, the smallest taking precedence.

iauCal2jd *Gregorian calendar to Julian Date* iauCal2jd

CALL :

```
j = iauCal2jd ( iy, im, id, &djm0, &djm );
```

ACTION :

Gregorian Calendar to Julian Date.

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 $0^{\rm h}$

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 any AD/BC numbering convention observed.

REFERENCE :

Seidelmann, P.K. (Ed.) (1992), *Explanatory Supplement to the Astronomical Almanac*, University Science Books, Section 12.92 (p604).

iauD2dtf *format for output a two-part JD* iauD2dtf

CALL :

```
j = iauD2dtf ( scale, ndp, d1, d2, iy, im, id, ihmsf );
```

ACTION :

Format for output a two-part Julian Date (or in the case of UTC a quasi-JD form that includes special provision for leap seconds).

GIVEN :

scale	`char`	time scale ID (Note 1)
ndp	`int`	resolution (Note 2)
d1,d2	`double`	time as a two-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`.

iauD2tf *days to hours, minutes, seconds* iauD2tf

CALL :

```
iauD2tf ( ndp, days, &sign, ihmsf );
```

ACTION :

Decompose days to hours, minutes, seconds, fraction.

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`s 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 `ihmsf[4]`.

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.

iauDat — *calculate TAI−UTC* — iauDat

CALL :

```
j = iauDat ( iy, im, id, fd, deltat );
```

ACTION :

For a given UTC date, calculate ΔAT = TAI−UTC.

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: +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 parameter `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 returned 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 *Explanatory Supplement to the Astronomical Almanac*, ed. P. Kenneth Seidelmann (1992), University Science Books.

iauDtdb *approximation to TDB−TT* iauDtdb

CALL :

```
iauDtdb ( date1, date2, ut, elong, u, v );
```

ACTION :

An approximation to TDB−TT, the difference between barycentric dynamical time and terrestrial time, for an observer on the Earth.

GIVEN :

date1	`double`	TDB as a two-part...
date2	`double`	...Julian Date (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 TDB date `date1+date2` is a Julian Date, apportioned in any convenient way between the arguments `date1` and `date2`. 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.

   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 routine, comprises a main (annual) sinusoidal term of amplitude approximately 0.00166s, plus planetary terms up to about 20μs, and lunar and diurnal terms up to 2μs. 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 1ns 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 routine 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 10^{-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 $(\mathbf{v}/c).(\mathbf{r}/c)$, where $\mathbf{v}$ is the barycentric velocity of the Earth, $\mathbf{r}$ 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 ±3ns 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 :

1. Fairhead, L., & Bretagnon, P., Astron.Astrophys., **229**, 240-247 (1990).
2. IAU 2006 Resolution 3.
3. McCarthy, D. D., Petit, G. (eds.), IERS Conventions (2003), IERS Technical Note No. 32, BKG (2004).
4. Moyer, T.D., Cel.Mech., **23**, 33 (1981).

5. Murray, C.A., *Vectorial Astrometry*, Adam Hilger (1983).

6. Seidelmann, P.K. (Ed.) (1992), *Explanatory Supplement to the Astronomical Almanac*, University Science Books, Chapter 2.

7. Simon, J.L., Bretagnon, P., Chapront, J., Chapront-Touzé, M., Francou, G. & Laskar, J., Astron.Astrophys., **282**, 663-683 (1994).

iauDtf2d *date and time fields to two-part JD* iauDtf2d

CALL :

```
j = iauDtf2d ( scale, iy, im, id, ihr, imn, sec, ihr, imn, sec, &d1, &d2 );
```

ACTION :

Encode date and time fields into two-part Julian Date (or in the case of UTC a quasi-JD form that includes special provision for leap seconds).

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`	two-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 (Note 3) −4 = bad minute −5 = bad second (<0)

NOTES :

1. The argument `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.

iauEpb *Julian Date to Besselian Epoch* iauEpb

CALL :

```
d = iauEpb ( dj1, dj2 );
```

ACTION :

Julian Date to Besselian Epoch.

GIVEN :

dj1,dj2	`double`	Julian Date (see note)

RETURNED (function value) :

`double`	Besselian Epoch

NOTE :

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.

iauEpb2jd *Besselian Epoch to Julian Date* iauEpb2jd

CALL :

```
iauEpb2jd ( epb, &djm0, &djm );
```

ACTION :

Besselian Epoch to Julian Date.

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`.

iauEpj — *Julian Date to Julian Epoch* — iauEpj

CALL :

```
d = iauEpj ( dj1, dj2 );
```

ACTION :

Julian Date to Julian Epoch.

GIVEN :

dj1,dj2	`double`	Julian Date (see note)

RETURNED (function value) :

`double`	Julian Epoch

NOTE :

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.

iauEpj2jd *Julian Epoch to Julian Date* iauEpj2jd

CALL :

```
iauEpj2jd ( epj, djm0, djm );
```

ACTION :

Julian Epoch to Julian Date.

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`.

iauGd2gc *geodetic to geocentric* iauGd2gc

CALL :

```
j = iauGd2gc ( n, elong, phi, height, xyz );
```

ACTION :

Transform geodetic coordinates to geocentric using the specified reference ellipsoid.

GIVEN :

n	`int`	ellipsoid identifier (Note 1)
elong	`double`	longitude (radians, east positive)
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 number `n` has no significance outside the SOFA software.

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.

iauJd2cal — *JD to year, month, day, fraction* — iauJd2cal

CALL :

```
j = iauJd2cal ( dj1, dj2, iy, im, id, fd );
```

ACTION :

Julian Date to Gregorian year, month, day, and fraction of a day.

GIVEN :

dj1,dj2	`double`	Julian Date (Notes 1, 2)

RETURNED :

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 10^9.

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 :

1. Seidelmann, P.K. (Ed.) (1992), *Explanatory Supplement to the Astronomical Almanac*, University Science Books, Section 12.92 (p604).
2. Klein, A., *A Generalized Kahan-Babuska-Summation-Algorithm*, Computing 76, 279-293 (2006), Section 3.

iauJdcalf *JD to calendar report fields* iauJdcalf

CALL :

```
j = iauJdcalf ( ndp, dj1, dj2, iymdf )
```

ACTION :

Julian Date to Gregorian Calendar, expressed in a form convenient for formatting messages: rounded to a specified precision, and with the fields stored in a single array.

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 = `ndp` not 0-9 (interpreted as 0) 0 = OK −1 = date out of range

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.

REFERENCE :

Seidelmann, P.K. (Ed.) (1992), *Explanatory Supplement to the Astronomical Almanac*, University Science Books, Section 12.92 (p604).

iauTaitt *TAI to TT* iauTaitt

CALL :

```
j = iauTaitt ( tai1, tai2, &tt1, &tt2 );
```

ACTION :

Time scale transformation: International Atomic Time, TAI, to Terrestrial Time, TT.

GIVEN :

tai1,tai2	`double`	TAI as a two-part Julian Date

RETURNED :

tt1,tt2	`double`	TT as a two-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 :

1. McCarthy, D. D., Petit, G. (eds.), *IERS Conventions (2003)*, IERS Technical Note No. 32, BKG (2004).
2. Seidelmann, P.K. (Ed.) (1992), *Explanatory Supplement to the Astronomical Almanac*, University Science Books.

iauTaiut1 *TAI to UT1* iauTaiut1

CALL :

```
j = iauTaiut1 ( tai1, tai2, dta, &ut11, &ut12 );
```

ACTION :

Time scale transformation: International Atomic Time, TAI, to Universal Time, UT1.

GIVEN :

tai1,tai2	`double`	TAI as a two-part Julian Date
dta	`double`	UT1−TAI in seconds

RETURNED :

ut11,ut12	`double`	UT1 as a two-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 :

Seidelmann, P.K. (Ed.) (1992), *Explanatory Supplement to the Astronomical Almanac*, University Science Books

iauTaiutc — *TAI to UTC* — iauTaiutc

CALL :

```
j = iauTaiutc ( tai1, tai2, &utc1, &utc2 );
```

ACTION :

Time scale transformation: International Atomic Time, TAI, to Coordinated Universal Time, UTC.

GIVEN :

tai1,tai2	`double`	TAI as a two-part Julian Date (Note 1)

RETURNED :

utc1,utc2	`double`	UTC as a two-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 the function `iauDat` for further details.

REFERENCES :

1. McCarthy, D. D., Petit, G. (eds.), *IERS Conventions (2003)*, IERS Technical Note No. 32, BKG (2004).
2. IAU 2000 Resolution B1.9.

iauTcbtdb *TCB to TDB* iauTcbtdb

CALL :

```
j = iauTcbtdb ( tcb1, tcb2, &tdb1, &tdb2 );
```

ACTION :

Time scale transformation: Barycentric Coordinate Time, TCB, to Barycentric Dynamical Time, TDB.

GIVEN :

tcb1,tcb2	`double`	TCB as a two-part Julian Date

RETURNED :

tdb1,tdb2	`double`	TDB as a two-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.

iauTcgtt — *TCG to TT* — iauTcgtt

CALL :

```
j = iauTcgtt ( tcg1, tcg2, &tt1, &tt2 );
```

ACTION :

Time scale transformation: Geocentric Coordinate Time, TCG, to Terrestrial Time, TT.

GIVEN :

tcg1,tcg2	`double`	TCG as a two-part Julian Date

RETURNED :

tt1,tt2	`double`	TT as a two-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 `tcg2` is the fraction of a day. The returned `tt1`, `tt2` follow suit.

REFERENCES :

1. McCarthy, D. D., Petit, G. (eds.), *IERS Conventions (2003)*, IERS Technical Note No. 32, BKG (2004).
2. IAU 2000 Resolution B1.9.

iauTdbtcb *TDB to TCB* iauTdbtcb

CALL :

```
j = iauTdbtcb ( tdb1, tdb2, &tcb1, &tcb2 );
```

ACTION :

Time scale transformation: Barycentric Dynamical Time, TDB, to Barycentric Coordinate Time, TCB.

GIVEN :

tdb1,tdb2	`double`	TDB as a two-part Julian Date

RETURNED :

tcb1,tcb2	`double`	TCB as a two-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.

iauTdbtt *TDB to TT* iauTdbtt

CALL :

```
j = iauTdbtt ( tdb1, tdb2, dtr, &tt1, &tt2 );
```

ACTION :

Time scale transformation: Barycentric Dynamical Time, TDB, to Terrestrial Time, TT.

GIVEN :

tdb1,tdb2	`double`	TDB as a two-part Julian Date
dtr	`double`	TDB−TT in seconds

RETURNED :

tt1,tt2	`double`	TT as a two-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 :

1. McCarthy, D. D., Petit, G. (eds.), *IERS Conventions (2003)*, IERS Technical Note No. 32, BKG (2004).
2. IAU 2000 Resolution 3.

iauTf2d *hours, minutes, seconds to days* iauTf2d

CALL :

```
j = iauTf2d ( s, ihour, imin, sec, &days );
```

ACTION :

Convert hours, minutes, seconds to days.

GIVEN :

s	`c`	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. If the `s` argument is a string, only the leftmost character is used and no warning status is provided.
2. The result is computed even if any of the range checks fail.
3. Negative `ihour`, `imin` and/or `sec` produce a warning status, but the absolute value is used in the conversion.
4. If there are multiple errors, the status value reflects only the first, the smallest taking precedence.

iauTttai — *TT to TAI* — iauTttai

CALL :

```
j = iauTttai ( tt1, tt2, &tai1, &tai2 );
```

ACTION :

Time scale transformation: Terrestrial Time, TT, to International Atomic Time, TAI.

GIVEN :

tt1,tt2	`double`	TT as a two-part Julian Date

RETURNED :

tai1,tai2	`double`	TAI as a two-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 :

1. McCarthy, D. D., Petit, G. (eds.), *IERS Conventions (2003)*, IERS Technical Note No. 32, BKG (2004).
2. Seidelmann, P.K. (Ed.) (1992), *Explanatory Supplement to the Astronomical Almanac*, University Science Books.

iauTttcg *TT to TCG* iauTttcg

CALL :

```
j = iauTttcg ( tt1, tt2, &tcg1, &tcg2 );
```

ACTION :

Time scale transformation: Terrestrial Time, TT, to Geocentric Coordinate Time, TCG.

GIVEN :

tt1,tt2	`double`	TT as a two-part Julian Date

RETURNED :

tcg1,tcg2	`double`	TCG as a two-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 :

1. McCarthy, D. D., Petit, G. (eds.), *IERS Conventions (2003)*, IERS Technical Note No. 32, BKG (2004).
2. IAU 2000 Resolution B1.9.

iauTttdb — *TT to TDB* — iauTttdb

CALL :

```
j = iauTttdb ( tt1, tt2, dtr, &tdb1, &tdb2 );
```

ACTION :

Time scale transformation: Terrestrial Time, TT, to Barycentric Dynamical Time, TDB.

GIVEN :

tt1,tt2	`double`	TT as a two-part Julian Date
dtr	`double`	TDB−TT in seconds

RETURNED :

tdb1,tdb2	`double`	TDB as a two-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 :

1. McCarthy, D. D., Petit, G. (eds.), *IERS Conventions (2003)*, IERS Technical Note No. 32, BKG (2004).
2. IAU 2000 Resolution 3.

iauTtut1 *TT to UT1* iauTtut1

CALL :

```
j = iauTtut1 ( tt1, tt2, dt, &ut11, &ut12 );
```

ACTION :

Time scale transformation: Terrestrial Time, TT, to Universal Time, UT1.

GIVEN :

tt1,tt2	`double`	TT as a two-part Julian Date
dt	`double`	TT−UT1 in seconds

RETURNED :

ut11,ut12	`double`	UT1 as a two-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 ΔT.

REFERENCE :

Seidelmann, P.K. (Ed.) (1992), *Explanatory Supplement to the Astronomical Almanac*, University Science Books.

iauUt1tai — *UT1 to TAI* — iauUt1tai

CALL :

```
j = iauUt1tai ( ut11, ut12, dta, &tai1, &tai2 );
```

ACTION :

Time scale transformation: Universal Time, UT1, to International Atomic Time, TAI.

GIVEN :

ut11,ut12	`double`	UT1 as a two-part Julian Date
dta	`double`	UT1−TAI in seconds

RETURNED :

tai1,tai2	`double`	TAI as a two-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 :

Seidelmann, P.K. (Ed.) (1992), *Explanatory Supplement to the Astronomical Almanac*, University Science Books.

iauUt1tt *UT1 to TT* iauUt1tt

CALL :

```
j = iauUt1tt ( ut11, ut12, dt, &tt1, &tt2 );
```

ACTION :

Time scale transformation: Universal Time, UT1, to Terrestrial Time, TT.

GIVEN :

ut11,ut12	`double`	UT1 as a two-part Julian Date
dt	`double`	TT−UT1 in seconds

RETURNED :

tt1,tt2	`double`	TT as a two-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 ΔT.

REFERENCE :

Seidelmann, P.K. (Ed.) (1992), *Explanatory Supplement to the Astronomical Almanac*, University Science Books.

iauUt1utc *UT1 to UTC* iauUt1utc

CALL :

```
j = iauUt1utc ( ut11, ut12, dut1, &utc1, &utc2 );
```

ACTION :

Time scale transformation: Universal Time, UT1, to Coordinated Universal Time, UTC.

GIVEN :

ut11,ut12	`double`	UT1 as a two-part Julian Date (Note 1)
dut1	`double`	ΔUT1: UT1−UTC in seconds (Note 2)

RETURNED :

utc1,utc2	`double`	UTC as a two-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. ΔUT1 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 the function `iauDat` for further details.

REFERENCES :

1. McCarthy, D. D., Petit, G. (eds.), *IERS Conventions (2003)*, IERS Technical Note No. 32, BKG (2004).
2. Seidelmann, P.K. (Ed.) (1992), *Explanatory Supplement to the Astronomical Almanac*, University Science Books.

iauUtctai *UTC to TAI* iauUtctai

CALL :

```
j = iauUtctai ( utc1, utc2, &tai1, &tai2 );
```

ACTION :

Time scale transformation: Coordinated Universal Time, UTC, to International Atomic Time, TAI.

GIVEN :

utc1,utc2	`double`	UTC as a two-part quasi Julian Date (Notes 1-4)

RETURNED :

tai1,tai2	`double`	TAI as a two-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 the function `iauDat` for further details.
4. The function `iauDtf2d` converts from calendar date and time of day into two-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.

REFERENCES :

1. McCarthy, D. D., Petit, G. (eds.), *IERS Conventions (2003)*, IERS Technical Note No. 32, BKG (2004).
2. Seidelmann, P.K. (Ed.) (1992), *Explanatory Supplement to the Astronomical Almanac*, University Science Books.

iauUtcut1 *UTC to UT1* iauUtcut1

CALL :

```
j = iauUtcut1 ( utc1, utc2, dut1, &ut11, &ut12 );
```

ACTION :

Time scale transformation: Coordinated Universal Time, UTC, to Universal Time, UT1.

GIVEN :

utc1,utc2	`double`	UTC as a two-part Julian Date (Notes 1-4)
dut1	`double`	ΔUT1: UT1−UTC in seconds (Note 2)

RETURNED :

ut11,ut12	`double`	UT1 as a two-part quasi 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 routine 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 the function `iauDat` for further details.
4. The function `iauDtf2d` converts from calendar date and time of day into two-part Julian Date, and in the case of UTC implements the leap-second-ambiguity convention described above.
5. ΔUT1 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 :

1. McCarthy, D. D., Petit, G. (eds.), *IERS Conventions (2003)*, IERS Technical Note No. 32, BKG (2004).
2. Seidelmann, P.K. (Ed.) (1992), *Explanatory Supplement to the Astronomical Almanac*, University Science Books.