CDS Time Conversion Software, and its application to STEREO

William Thompson
October 24, 2005

1 Introduction

This document describes a time-handling package within SolarSoft. It was originally written for the CDS instrument on SOHO, but is used in a number of places within SolarSoft. In particular, it's used within some of the software developed for the STEREO project, such as the SPICE routines, and the telemetry parsing routines.

2 Time formats

The primary impetus for the development of this software was the need to distinguish between International Atomic Time (TAI), and Coordinated Universal Time (UTC). The SOHO spacecraft uses TAI internally, but all the science planning is done in the more usual UTC.

The time maintained internally in the STEREO project is formatted similarly to the SOHO on-board time, but is based on UTC rather than TAI. Some additional features have been added to the software to take this into account.

2.1 TAI—International Atomic Time

TAI is defined as the number of standard seconds since 0h on 1 January 1958. In the CDS software, TAI time is always expressed as a double precision floating point number.

Although the STEREO spacecraft does not keep TAI internally, the concept of TAI is still very useful. TAI time always increases monotonically, and the duration of any event can be calculated as the difference between the start and end times expressed in TAI. The situation with UTC is more complicated when leap seconds need to be taken into account. By expressing times as TAI double-precision numbers, one can do time calculations without worrying about leap seconds. The result can then be easily converted back to UTC.

2.2 UTC—Coordinated Universal Time

This is the time standard on which civil time is based. The main distinction between UTC and TAI, at least since 1 January 1972, is that occasionally a "leap second" is inserted into the UTC time to keep it in sync with the rotation of the earth. (Before 1972 the situation was more complicated.) TAI time has no leap seconds. Therefore, in order to convert between the two kinds of time, one needs to know when leap seconds were added to the UTC time. This information is maintained within the file "leap_seconds.dat" in the directory given by the environment variable TIME_CONV. This file is delivered as part of SolarSoft, and the default value of TIME_CONV is defined as part of the SolarSoft setup.

Some operating systems, such as VMS and older versions of MacOS, keep track only of local time, not UTC. In such cases, one can store the difference in hours (local-UTC) in the text file "local_diff.dat" in the same TIME_CONV directory as "leap_seconds.dat" above. For example, for U.S. Eastern Standard Time, this file would contain the value -5.

If the computer is running on GMT rather than local time, one can signify this by adding a second line to the "local_diff.dat" file with the letters "GMT". For example, if one is on the east coast of the United States during winter, and the computer is set up to use GMT, then the file would contain the lines

```
-5
GMT
```

When daylight saving time is in effect this would need to be changed to reflect this.

It is not necessary for "leap_seconds.dat" and "local_diff.dat" to be in the same directory. One can define TIME_CONV to be a set of directories, using the same format one would use for IDL_PATH. That way, one can have a single "leap_seconds.dat" file which is common among a number of computers (e.g. through NFS or mirror), and a separate "local_diff.dat" file for each computer.

Most modern computers do not require a "local_diff.dat" file. In most cases, the automatic definition of TIME_CONV within SolarSoft will suffice.

Note that if time differences between observations are needed to an accuracy of a second and the observations are separated by more than a day then the UTC times should be converted to TAI before differencing in case there is an intervening leap-second.

There are three formats that the CDS software uses for UTC, all of which are calendar-based. These are:

Internal: Referred to as "INT" in any routine names. A structure containing the following elements as longword integers:

- MJD — The Modified Julian Day (MJD) number. This is defined as the ordinary Julian Day (JD) number minus 2400000.5. The ".5" represents the fact that MJD numbers begin at midnight, whereas JD numbers begin at noon.
- TIME — The time of day, in milliseconds since the beginning of the day.

External: Referred to as "EXT" in any routine names. A structure containing the elements, YEAR, MONTH, DAY, HOUR, MINUTE, SECOND, and MILLISECOND as shortword integers.

String: Referred to as "STR" in any routine names. A calendar date in ASCII string format. This format is subdivided into the following subcategories.

CCSDS: A string variable containing the calendar date in the format recommended by the Consultative Committee for Space Data Systems, which in turn is based on ISO 8601, e.g.

"1988-01-18T17:20:43.123"

ECS: A variation on the CCSDS format used by the SOHO EOF Core System. The "T" separator is eliminated, and slashes are used instead of dashes in the date, e.g.

"1988/01/18 17:20:43.123"

VMS: Similar to that used by the VMS operating system, this format uses a three-character abbreviation for the month, and rearranges the day and the year, e.g.

"18-JAN-1988 17:20:43.123"

STIME: Based on !STIME in IDL, this format is the same as the VMS format, except that the time is only given to 0.01 second accuracy, e.g.

"18-JAN-1988 17:20:43.12"

Other string types may be added in the future.

3 Procedures

The main procedures are ANYTIM2UTC and ANYTIM2TAI, which take a wide range of inputs, and convert them into one of the standard formats discussed in the previous section. The default format for ANYTIM2UTC is CDS internal format, but one can use keywords to select the CDS external format, or one of the standard string formats. ANYTIM2TAI always returns the double precision number of seconds since 1-Jan-1958.

Another useful routine is ANYTIM2CAL, which outputs some calendar time formats not otherwise supported by the software.

3.1 Underlying procedures

The following procedures are used to convert between the different time formats:

OBT2TAI	Converts the 6-byte SOHO on-board time (OBT/LOBT) to TAI format. *(Not used by STEREO).*
TAI2UTC	Converts TAI times to any one of the CDS UTC formats.
UTC2TAI	Converts any one of the CDS UTC formats to TAI.
INT2UTC	Converts internal time to either external or CCSDS format.
UTC2INT	Converts either external or CCSDS time to internal format.
STR2UTC	Converts string time to either internal or external format.
UTC2STR	Converts either internal or external time to string format.

In addition, there are also the following routines:

UTC2DOW	Calculates the day of the week from any of the UTC formats.
UTC2DOY	Calculates the day of the year from any of the UTC formats.
GET_UTC	Gets the current UTC date/time from the system clock in any one of the CDS UTC formats.
CHECK_INT_TIME	Checks the internal time for consistency.
CDS2JD	Calculate full Julian day equivalent of CDS date/time.
LOCAL_DIFF	Returns the difference in hours between local and UTC time.

The following are essentially internal routines:

DATE2MJD	Converts dates to MJD numbers.
MJD2DATE	Converts MJD numbers to dates.
GET_LEAP_SEC	Gets the MJD numbers for days with leap seconds.

The following are used in the UTPLOT programs and take no account of leap seconds:

UTC2SEC	Converts CDS UTC time format to seconds since MJD=0.
SEC2UTC	Converts seconds since MJD=0 to CDS UTC format.

Note that where a routine uses a particular form of time, any of the specified formats for that time may be used as input to the routine–the code can distinguish the formats. Thus, in the routine STR2UTC the input variable may be any recognized string format, including CCSDS or ECS, plus some variations on these. In UTC2TAI the input can be any of the UTC formats mentioned above (INT, EXT, STR).

If there is a choice on output, then this is controlled by keywords. For instance, when using INT2UTC the keywords /EXTERNAL, /CCSDS or /ECS would be used to determine the format of the UTC output.

By default, the software always assumes that any double precision number is a TAI time. However, the keyword /NOCORRECT was added to *some* of the routines to support STEREO on-board times, which are UTC-based. When /NOCORRECT is used, the software assumes that the time is already in the target time system. Support for this convention is limited—users are encouraged to parse the telemetry time parameters in one of the standard UTC formats, using TAI2UTC(/NOCORRECT), and follow the normal UTC/TAI conventions from then on.

Appendices

A Notes on string formats

The two supported string date/time formats, CCSDS and ECS, are very similar in the way that they represent the date. Each uses a four digit year, followed by a numerical month, and then a day of the month. There are certainly other ways to represent dates. For example, some common representations are:

January 18, 1988
18-JAN-1988
01/18/88
18/01/88

There are a number of reasons to use the CCSDS/ECS style scheme in preference to any of the above variations. Some of these reasons are as follows:

- In the United States it is popular to use a month-day-year format, while in Europe a day-month-year format is more common. Thus, a date such as 01/02/95 could be interpreted as either January 2, 1995 or as 1 February 1995. Using instead a year-month-day format eliminates this source of confusion.
- Starting the date off with a full four-character year makes it self-evident what the date convention is. A date such as 01/02/03 could be interpreted as January 2, 2003, as 1 February 2003, or as 3 February 2001.
- Using numerical months instead of character months, together with a year-month-day convention, means that dates sorted by date are also sorted alphabetically, and vice-versa. This can be useful in software development.

The CDS string parsing routine STR2UTC is more flexible in the kinds of strings that it accepts than UTC2STR is in formatting them. For example, it can accept strings such as "88/01/18". However, it is very conservative in how it attempts to guess what order is used for the year, month, and day. Normally, it assumes that the date is in the format year-month-day. Automatic detection of the two other possibilities, day-month-year or month-day-year, is supported only if the month is given as a character string. For example, the string "18-Jan-1988" is automatically recognized. Both the VMS and STIME formats satisfy the above conditions, so they have been added as officially supported formats. In other cases, the day-month-year and month-day-year conventions are supported through the keywords /DMY and /MDY.

Note: Due to popular demand, it was decided that the STR2UTC routine should assume that when the month is given as a character string, then the year is the *last* parameter in the date, and not the first as before. Thus, the software will automatically recognize dates of the form "18-Jan-88", *but will not automatically support dates such as "88-Jan-18"*. There are two ways that this can be overcome—i.e. two ways that dates with a character month and beginning with a year can be supported:

1. Give the year with all four digits.

2. Use the keyword /YMD.

Generally speaking, it is safest to always use a four digit year.

The following table illustrates various date formats which are supported. It is not meant to be an exhaustive list.

1995-02-15	
1995/02/15	
95/02/15	
1995-046	(Day-of-year variation)
15-Feb-95	
1995-February-15	
15-FEB-1995	
02/15/95	(Only with /MDY keyword)
15/02/95	(Only with /DMY keyword)
95-Feb-15	(Only with /YMD keyword)