# astrolib 1.03

The astrolib library calculates planetary positions and time zones.

Positions of sun, moon and planets up to Pluto are calculated with an accuracy of the order of one arc minute for dates less than a century away from now. Calculations are based on approximation formulae by Paul Schlyter, which are in turn based on a preprint of a paper by T. van Flandern and K. Pulkkinen "Low precision formulae for planetary positions", Astrophysical Journal Supplement Series, 1980.

Information is retrieved about locations and time zones on earth, based on UNIX time zone data. The data contains information for over 350 locations world-wide and is even useful for dates centuries before UNIX machines existed. The same information is provided for about 2000 additional locations which have been assigned to UNIX time zone locations.

Currently, astrolib is implemented as a shared library for Palm OS 2.0 or later. It requires the MathLib library. Because the time zone data has been compressed quite a bit and the calculations do not require any external ephemeris data files, the library is only about 100K.

The astrolib source code is distributed under the GNU Lesser General Public License. As I understand the license, this means essentially that you can use astrolib from commercial and non-commercial software, but that you cannot make a commercial product out of a modified version of astrolib. Redistribution under the GNU LGPL is, of course, possible.

Click here to download the source packaged for Mac or Windows. astrolib has been developed on a Mac using CodeWarrior. If you want to build using CodeWarrior on Windows or using the GNU compiler, you will have to adapt the projects and/or the source. Read section 2 for additional info.

## 1. How to use astrolib

The interface of astrolib has been designed to hide as much functional details of the implementation as possible from the user while still providing all the required information. Care has been taken to make the interface independent of any specific device or operating system and to make it as forward compatible as possible.

This section provides a typical example of how to use astrolib. The interface is specified directly in astrolib's header file "astrolib.h". Additional examples can be found in "alibtest.c".

Add "astrolib.h" and "astrolib.c" to your project. Then start using astrolib as follows:

```
aLibErr err;
Int16 version;
version = aLibVersion;
err = aLibNew(&version);
// handle error
```

Now you are ready to use astrolib. Here are some examples that retrieve information about places on earth:

```
Int16 i;
```

```
Char a[99];
err = aLibMapGetItem(mapCountries,0,&i); // 237
err = aLibMapGetItem(mapLocationName,100,a); // "Douala"
err = aLibMapGetItem(mapLocationArea,100,&i); // 0
err = aLibMapGetItem(mapAreaName,i,a); // "Africa"
err = aLibMapGetItem(mapLocationCountry,100,&i); // 35
err = aLibMapGetItem(mapCountryName,i,a); // "Cameroon"
err = aLibMapGetItem(mapCountryCode,i,a); // "CM"
err = aLibMapGetItem(mapLocationLon,100,&lon); // 0.16930 = 9e42
err = aLibMapGetItem(mapLocationInfo,77,a); // "China mountains"
```

You can search areas, countries and locations by name:

```
err = aLibMapFindByName(mapLocationName,"Z\xFCrich",&i); // 365
err = aLibMapGetItem(mapLocationName,i,a); // "Zurich"
```

Here is how to determine time zones:

```
err = aLibMapFindByName(mapLocationName,"Istanbul",&i);
year=1978; month=10; day=15; hour=23; min=59;
err = aLibTimezone
(year,month,day,hour,min,0,calGregorian,relLocal,i,
0,&offsStandard,&offsLocal,zoneName);
// offsStandard=2, offsLocal=3, zoneName="EEST"
```

For astronomical calculations you need to specify time and place. Let's determine the Julian Day for 19 April 1990, 0:00 UTC:

```
double jd;
err = aLibTimezone(1990,4,19,0,0,0,calGregorian,relUTC,0,&jd,0,0,0);
// jd = 2448000.5
```

Then create an event by indicating time and place:

```
aLibEvent e;
lon=15.0/(180/pi);
lat=60.0/(180/pi);
err = aLibEventNew(&e,jd,lon,lat);
// handle error
```

Now you are ready to do some astronomical calculations:

```
double o[3];

// get obliquity of the ecliptic
err = aLibEventGetItem(e,eventEcl,&ecl);
// 0.40911 = 23.4406 deg

// get geocentric ecliptical position of the Sun
err = aLibEventGetObjectItem (e,objSun,objCoordsSpheGeoEcl,o);
// lon = 0.50068 = 28.6870 deg, lat = 0, r = 1.004323

// get topocentric equatorial position of the moon
err = aLibEventGetObjectItem (e,objMoon,objCoordsSpheTopoEqu,o);
// RA = 5.41055 = 310.0019 deg, Decl = -0.34696 = -19.8795 deg

// get geocentric equatorial position of Mercury
err = aLibEventGetObjectItem (e,objMercury,objCoordsSpheGeoEqu,o);
```

```
// RA = 0.75503 = 43.2599 deg, Decl = 0.34289 = 19.6495 deg, r =
0.748296 (AU)
```

When you are done using astrolib, free any created events and then astrolib:

```
aLibEventFree(&e);
aLibFree();
```

# 2. Building astrolib

This section describes first how to build astrolib on a Mac using Metrowerks CodeWarrior, then gives some information about building on or for different systems.

astrolib consists of 3 CodeWarrior projects: "tzgen", "alibtest" and "astrolib".

### 2.1 tzgen

This project is a console application that targets your desktop machine. It extracts and compresses UNIX time zone data files. It produces 3 text files in the "generated" folder. The file "world" is just a concatenation without comments of all zones, rules and links. The file "tzdata.h" defines a few constants needed in astrolib. The file "tzdata.txt" contains the actual compressed data.

This is the output of "tzgen" in verbose mode for tzdata2000g:

```
Creating data structures from timezone data...
Step 1: Reading countries from file 'iso3166.tab'... 239 countries
read.
Step 2: Reading areas from file 'zone.tab'... 10 areas read.
Step 3: Reading locations from file 'zone.tab'... 366 locations read.
Step 4: Removing countries without locations...
Removed country 27 BV Bouvet Island
Removed country 93 HM Heard Island & McDonald Islands
2 countries removed.
Step 5: Concating continents to file 'world'... 7 files concated
(3226 lines).
Step 6: Reading format strings from file 'world'...
298 strings read.
Step 7: Reading rules from file 'world'...
125 rules read (1631 items).
Maximal number of items per rule: 74.
Step 8: Reading zones from file 'world'...
Could not add zone: could not split location Zone WET 0:00 EU
WE%sT...
Could not add zone: could not split location Zone CET 1:00 C-Eur
CE%sT...
Could not add zone: could not split location Zone MET 1:00 C-Eur
ME%sT...
Could not add zone: could not split location Zone EET 2:00 EU
EE%sT...
356 zones read (1570 items).
Maximal number of items per zone: 14.
```

```
Step 9: Reading links from file 'world'...
Link with non-existing link location: Europe/Nicosia...
Link with non-existing link location: Asia/Istanbul...
Link with non-existing link location: America/Indiana/Indianapolis...
Link with non-existing link location: America/Kentucky/Louisville...
Link with non-existing link area: EST5EDT...
Link with non-existing link area: CST6CDT...
Link with non-existing link area: MST7MDT...
Link with non-existing link area: PST8PDT...
Link with non-existing link area: EST...
Link with non-existing link area: MST...
Link with non-existing link area: HST...
10 links read.
Step 10:Reading locations2 from file 'moreplaces'... 2120 locations
read.
Checking data consistency... OK.
Linking locations of same country... OK.
Exporting header file... done.
Exporting serialized data...
country: 2860 ( 22874 bits)
area: 72 ( 574 bits)
location: 9188 ( 73500 bits)
rule: 11668 ( 93337 bits, max=4201 bits)
format: 1398 ( 11177 bits)
zone: 13970 (111759 bits, max= 920 bits)
location2:34692 (277536 bits)
73848 bytes (72k) exported.

Finished.
```

### 2.2 alibtest

This project targets the Palm and is intended for testing astrolib. It can be operated in two modes: either by accessing astrolib's functions by loading the shared library or else by directly linking the functions into "alibtest". The latter mode has the advantage that one can step into astrolib's functions in the CodeWarrior debugger. The internal mode is activated by defining ALIB_INTERNAL_TEST in "myprefix.h".

### 2.3 astrolib

This projects builds astrolib. To update the time zone data, copy the compressed data generated by "tzgen" from "tzdata.txt" to the resource file "astrolib.rsrc". To do this, open "tzdata.txt" in any text editor, select and copy all lines and paste into resources ID 9000-9006 of type "aLib" using ResEdit.

### 2.4 Compiling on/for different systems

The most similar system is CodeWarrior on Windows. You will have to see for yourself whether

the project files are cross-platform. If not, you will have to create new ones from scratch.

In any case, you will have to recreate astrolib's resources. Other than that, you will probably be able to build astrolib with little or no changes to the source.

I cannot offer much help regarding a port to the GNU compiler. What I can say is that the "tzgen" project should not make any problems as it is a simple console application; building the Palm OS projects however might not be easy and might require changes to the source.

Finally, porting astrolib to other devices and/or operating systems is potentially attractive. The interface has nothing Palm specific, so it should be implementable on practically any device with a C compiler.

# 3. License



# 4. Version history

May-Oct 2000
Version 1.0 by delphi29@excite.com

Oct 2000
Version 1.01 by delphi29@excite.com
- fixed bug in version comparison when opening astrolib
- fixed bug in timezone calculation
- tried to clarify the documentation

Oct 2000
Version 1.02 by delphi29@excite.com
- updated time zone source data to tzdata2000g
- fixed bug that read from unlocked memory after closing astrolib

May 2001
Version 1.03 by delphi29@excite.com
- 2000 additional locations, each assigned to a timezone location
- faster and flashable (pointer to global data is now stored in A4 register, no longer in own resources)
- quick access to all locations in a country

# 5. Links

This file: http://jove.prohosting.com/~delphi29/astrolib.html