Micromega Corporation

Application Note 38

Calculating Sunrise and Sunset Times

Introduction

This application note describes how the uM-FPU V3.1 floating point coprocessor can be used to calculate sunrise and sunset times. The sunrise and sunset calculations are based on the algorithms and formulas published in the:

Almanac for Computers, 1990
Nautical Almanac Office
United States Naval Observatory

An explanation of sunrise and sunset calculations can be found on Ed Williams Aviation Page. Ed Williams' website also has an extensive set of formulas and explanations for navigational calculations. It's located at:

http://williams.best.vwh.net/

Sunrise and Sunset Calculation

Various sources on the web provide detailed explanations of the calculations required to determine sunrise and sunset times for a given location, as well as background on celestial coordinate systems and calculations. The algorithm used by the example described by this application note can be can be summarized as follows:

- **Calculate the Day of the Year**
  The year, month and day values are used to calculate the day of the year, including any adjustment for leap years.

- **Convert Longitude to Hours**
  The Earth rotates through 360 degrees in 24 hours, so 15 degrees of longitude will transit past a given meridian each hour. Longitude can be represented as an hour value using this conversion.

- **Calculate the Sun's Mean Anomaly**
  The Sun's mean anomaly determines the Sun's position on a circular orbit rather than the true elliptical orbit.

- **Calculate the Sun's Right Ascension and Declination**
  The equatorial coordinate system is a system used for mapping the position of objects in the sky. This system projects the object onto an imaginary sphere, called the celestial sphere. The coordinates of this system are Right Ascension (the celestial equivalent of Longitude) and Declination (the celestial equivalent to Latitude).

- **Calculate the Sun's Local Hour Angle**
  The Sun's hour angle is determined from the latitude and the zenith value for the type of sunrise or sunset time being calculated. The different types are described below.

- **Calculate the Local Mean Time of Sunset or Sunset**
  The UTC time of the sunset or sunrise is calculated using the hour angle, right ascension and longitudinal adjustment.

- **Adjust for Local Time**
  A time zone offset is used to adjust for local time.

There are four different sunrise and sunset definitions in common use: official, civil, nautical, and astronomical. Of these, the most commonly used definition is the official sunrise and sunset which occurs when the upper edge of the Sun is on the horizon. But even when the Sun is below the horizon, twilight can still provide illumination, so other definitions are used for special purposes. Civil sunrise and sunset is defined as when the center of the Sun is 6 degrees below the horizon. Nautical sunrise and sunset is defined as when the center of the Sun is 12 degrees below the horizon. Astronomical sunrise and sunset is defined as when the center of the Sun is 18 degrees below the horizon.

Values for `year`, `month`, `day`, `timeZone`, `lat1` and `long1` are first stored in the uM-FPU registers. The `sunTime` user-defined function is then called with a value in register 0 that specifies the type of calculation to perform. The register values are used to calculate the sunrise or sunset time for the specified location using the algorithm described above. The result is returned in register 0 as a 32-bit floating point number that specifies decimal hours in floating point. For example, a return value of 10.25 hours corresponds to 10 hours and 15 minutes. A conversion routine is provided to convert the decimal hour value to a readable text string (HH:MM).

sunTime Demonstration Program

The *sunTime* demonstration program requires no external connections. It uses constant values defined in the program to set the date, longitude and latitude. It then calculates and displays the sunrise and sunset times for the values specified. You can modify this program to try different dates and positions. Figure 1 shows sample output from the program.

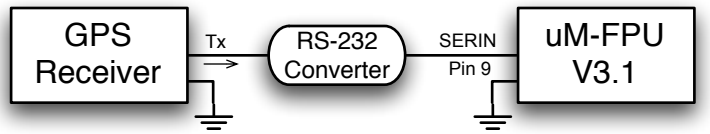
Figure 1: Sample Output from sunTime

```
Sunrise/Sunset Times
uM-FPU V3.1.0

Date:       24-Jul-07
Time Zone:  -4
Latitude:   44.23333,  44∞14'0.0"
Longitude: -76.50000, -76∞30'0.0"

Official sunrise:      5:45, sunset: 20:40
Civil sunrise:         5:11, sunset: 21:14
Nautical sunrise:      4:27, sunset: 21:57
Astronomical sunrise:  3:36, sunset: 22:48

Done.
```

sunTimeGPS Demonstration Program

The *sunTimeGPS* demonstration program reads the current date, time, longitude and latitude position from a GPS receiver. It then calculates and displays the sunrise and sunset times for the current location and time. It requires the serial output of a GPS receiver to be connected to the `SERIN` pin on the uM-FPU as shown below.

Figure 2: Connecting a GPS Receiver to the uM-FPU V3.1 chip



The serial output from the GPS receiver must be an NMEA data interface running at 4800 baud. Figure 3 shows sample output from the program.

Figure 3: Sample Output from sunTimeGPS

```
Sunrise/Sunset Times for GPS location
uM-FPU V3.1.0

Time Zone:  -4
Reading GPS data...

Date/Time:  24-Jul-07 14:59:54
Latitude:   44.25893,  44∞15'32.2"
Longitude: -76.37440, -76∞22'27.8"

Official sunrise:      5:44, sunset: 20:39
Civil sunrise:         5:10, sunset: 21:13
Nautical sunrise:      4:26, sunset: 21:57
Astronomical sunrise:  3:35, sunset: 22:48

Done.
```

sunTime.fpu Functions

The *sunTime.fpu* file contains uM-FPU V3.1 user-defined functions to calculate sunrise and sunset times, convert decimal hours to a text string, convert date and time values to strings, and to read data from a GPS receiver. A summary of each uM-FPU function is shown below.

sunTime

The sunTime function calculates the sunrise or sunset time for the given location and returns the result as decimal hours. If sunrise or sunset does not occur on the date and location specified, a value of -1.0 is returned.

Input:

register 0	32-bit integer	type of calculation to perform 0 - Official Sunrise 1 - Official Sunset 2 - Civil Sunrise 3 - Civil Sunset 4 - Nautical Sunrise 5 - Nautical Sunset 6 - Astronomical Sunrise 7 - Astronomical Sunset
year	32-bit integer	year (0 to 99)
month	32-bit integer	month of year (1 to 12)
day	32-bit integer	day of month (1 to 31)

timeZone	32-bit integer	hours of adjustment from UTC
lat1	32-bit float	latitude in degrees
long1	32-bit float	longitude in degrees

Output:

register 0	32-bit float	decimal hours (e.g. 10.25 = 10 hours, 15 minutes)

insertHours

Converts decimal hours to a text string and stores it at the current string selection point. The format of the string is *HH:MM*.

Input:

register 0	32-bit float	decimal hours

Output:

string selection	string	*HH:MM* e.g. `05:35`

insertDigits

Converts an integer value to a two-digit string and stores it at the current string selection point. If the value is less than 10, a leading zero is stored.

Input:

register 1	32-bit integer	0 to 99

Output:

string selection	string	*nn* e.g. `19`

insertDate

Converts year, month, and day integer values to a date string and stores it at the current string selection point. The format of the string is *DD-mmm-YY*.

Input:

year	32-bit integer	year (0 to 99)
month	32-bit integer	month of year (1 to 12)
day	32-bit integer	day of month (1 to 31)

Output:

string selection	string	*DD-mmm-YY* e.g. `23-Jul-07`

insertTime

Converts hour, minute and second integer values to a time string and stores it at the current string selection point. The format of the string is *HH:MM:SS*.

Input:

hour	32-bit integer	year (0 to 23)
minute	32-bit integer	month of year (0 to 59)
second	32-bit integer	day of month (0 to 59)

Output:

string selection	string	*HH:MM:SS* e.g. `10:40:45`

insertDateTime

Converts year, month, day, hour, minute and second integer values to a date/time string and stores it at the current string selection point. The format of the string is *DD-mmm-YY HH:MM:SS*.

Input:

hour	32-bit integer	year (0 to 23)
minute	32-bit integer	month of year (0 to 59)
second	32-bit integer	day of month (0 to 59)

Output:

string selection	string	*DD-mmm-YY HH:MM:SS* e.g. `23-Jul-07 10:40:45`

localTime

Adds the time zone offset to the current time. If the resulting time is in the previous or next day, the day, month and year vales are adjusted accordingly.

Input:

year	32-bit integer	year (0 to 99)
month	32-bit integer	month of year (1 to 12)
day	32-bit integer	day of month (1 to 31)
hour	32-bit integer	year (0 to 23)

Output:

year	32-bit integer	year (0 to 99)
month	32-bit integer	month of year (1 to 12)
day	32-bit integer	day of month (1 to 31)
hour	32-bit integer	year (0 to 23)

monthDays

Returns the number of days in the month and year specified. An adjustment for leap years is included.

Input:

year	32-bit integer	year (0 to 99)
month	32-bit integer	month of year (1 to 12)

Output:

register 0	32-bit integer	days in month

insertDegrees

Converts a value in degrees to a string and stores it at the current string selection point. The format of the string is *DDDD°MM'SS.S"*.

Input:

lat1	32-bit float	latitude in degrees
long1	32-bit float	longitude in degrees

Output:

string selection	string	*DDDD°MM'SS.S"* e.g. `-76°30'0.0"`

readNMEA

This function sets the `SERIN` pin for NMEA input at 4800 baud. It then waits for a valid NMEA sentence, checks for a GPRMC sentence, confirms that the status field indicates valid data, then calls `parseGPRMC` to extract the date, time, latitude and longitude information. It waits indefinitely until a valid GPRMC sentence is received.

Input:

none

Output:

year	32-bit integer	year (0 to 99)
month	32-bit integer	month of year (1 to 12)
day	32-bit integer	day of month (1 to 31)

lat1	32-bit float	latitude in degrees
long1	32-bit float	longitude in degrees

parseGPRMC

This function parses a GPRMC sentence and extracts the date, time, latitude and longitude information. The date is stored in the `year`. `month` and `day` registers. The UTC time is adjusted for the local time zone and stored in the `hour`, `minute` and `second` registers. Latitude and longitude are converted from NMEA format to decimal degrees and stored in the `lat1` and `long1` registers.

Input:

string buffer	NMEA sentence	

Output:

year	32-bit integer	year (0 to 99)
month	32-bit integer	month of year (1 to 12)
day	32-bit integer	day of month (1 to 31)
lat1	32-bit float	latitude in degrees
long1	32-bit float	longitude in degrees

NMEA_Degrees

This function converts the NMEA format for latitude and longitude (i.e. DDDMM.MMMM) to a decimal degree value suitable for calculations (i.e. DDD.DDDDD).

Input:

string buffer	NMEA sentence	
register 1	32-bit integer	string field number of the latitude or longitude field

Output:

register 0	32-bit float	latitude or longitude in degrees

Additional Files

There are additional files located on the Micromega website that accompany this application note. They include:

sunTime.fpu	contains uM-FPU V3.1 user-defined functions
sunTime.bs2	Basic Stamp application to print sunrise and sunset times from a fixed input
sunTimeGPS.bs2	Basic Stamp application to print sunrise and sunset times for the current GPS location
sunTime.bas	PICAXE application to print sunrise and sunset times from a fixed input
sunTimeGPS.bas	PICAXE application to print sunrise and sunset times for the current GPS location

Before running the demo application, the user-defined functions in *sunTime.fpu* must be programmed to the uM-FPU V3.1 chip. This can be done using the uM-FPU V3 IDE software.

Further Information

See the Micromega website (http://www.micromegacorp.com) for additional information regarding the uM-FPU V3.1 floating point coprocessor, including:

uM-FPU V3.1 Datasheet
uM-FPU V3.1 Instruction Set
Using the uM-FPU V3 Integrated Development Environment (IDE)
Application Note 36 - Reading GPS Data