Micromega Corporation

# Application Note 39

# Calculating Great Circle Distances

## Introduction

This application note describes how to calculate the great circle distance between two points on the surface of the Earth. A demonstration program and uM-FPU V3.1 user-defined functions implement the necessary calculations. Since the majority of the work is done directly on the FPU chip, very little overhead is required on the microcontroller. This makes it possible to read GPS data and implement great circle distance calculations and other navigation calculations on a wide range of microcontrollers from low-end to high-end.

The equation used for calculating great circle distances can be found on the Ed Williams Aviation Page. Ed Williams' website also has an extensive set of formulas and explanations for other navigational calculations. The website is located at:

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

## Great Circle

For navigation purposes, a great circle is defined as a circle that has the same circumference as the sphere of the Earth, with a center that is the same as the spherical center. When a great circle is used to plot a path between two points on the Earth's surface, it represents the shortest path between the two points, when traveling along the surface. The great circle route is particularly useful for long distance aviation and nautical routes, although many other factors will usually contribute to the actual route chosen.

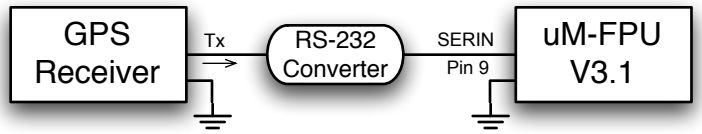
## Latitude and Longitude

Latitude and longitude is represented by the NMEA data interface as DDDMM.MMM. In order to perform navigational calculations these values must be converted to degrees or radians. The demonstration program included with this application note stores latitude and longitude values in radians so they can be used directly by the trigonometric functions. The sign os the values is positive for East and North values, and negative for West and South values.

## *GCdistance Demonstration Program*

The *GCdistance* program displays the great circle distance from an initial location to a series of pre-defined locations stored in user-defined functions on the uM-FPU V3.1 chip. The initial location can either be selected as one of the pre-defined locations, or the current latitude and longitude can be read from a GPS receiver. By default, the program will use initial location 0 (London, England). Other pre-defined locations can be used by changing the `from` variable to a value in the range 0 to 7. If the `from` variable is set to 99, the current location will be read from a GPS receiver connected to the SERIN pin as shown below.

**Figure 1: 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 2 shows sample output from the program.

**Figure 2: Sample Output from GCdistance**

```
Great Circle Distance
uM-FPU V3.1.0

From:
  London, England (51˚32', 0˚5')
To:
  Paris, France (48˚49', 2˚20')
      188.45 nautical miles
      216.86 miles
      349.01 kilometers
  Sydney, Australia (-34˚1', 151˚0')
     9166.71 nautical miles
    10548.86 miles
    16976.75 kilometers
  Tokyo, Japan (35˚41', 139˚45')
     5158.02 nautical miles
     5935.74 miles
     9552.65 kilometers
  Zurich, Switzerland (47˚22', 8˚31')
      418.56 nautical miles
      481.67 miles
      775.18 kilometers
  New York, USA (40˚47', -73˚58')
     3002.69 nautical miles
     3455.44 miles
     5560.99 kilometers
  Los Angeles, USA (34∞4', -118∞15')
     4724.95 nautical miles
     5437.37 miles
     8750.60 kilometers
  Santiago, Chile (-33˚29', -70˚45')
     6304.11 nautical miles
     7254.64 miles
    11675.21 kilometers
Done.
```

## *GCdistance.fpu* Functions

The *GCdistance.fpu* file contains uM-FPU V3.1 user-defined functions to do the following:

- calculate the great circle distance between two locations
- load information for pre-defined locations
- convert latitude and longitude values to text strings
- read data from a GPS receiver

A summary of each uM-FPU function is shown below.

### getID

Returns an ID number that the main program can use as a simple check to confirm that the correct set of user-defined functions have been programmed on the uM-FPU V3.1 chip. Register 0 is set to zero before calling this function. If no functions have been programmed it will remain zero. If the correct `getID` function is programmed, a value of 39 is returned.

*Input*:

| | | |
|---|---|---|
| register 0 | 32-bit integer | 0 |

*Output*:

| | | |
|---|---|---|
| register 0 | 32-bit integer | 39 |

### getDistance

Calculates the great circle distance between the two points specified.

*Input*:

| | | |
|---|---|---|
| lat1 | 32-bit float | latitude of first point (radians) |
| long1 | 32-bit float | latitude of first point (radians) |
| lat2 | 32-bit float | latitude of second point (radians) |
| long2 | 32-bit float | latitude of second point (radians) |

*Output*:

| | | |
|---|---|---|
| register 0 | 32-bit float | distance (nautical miles) |

### getLocation

Loads the latitude, longitude and name for the specified location.

*Input*:

| | | |
|---|---|---|
| register 0 | 32-bit integer | location index<br>0 - London, England<br>1 - Paris, France<br>2 - Sydney, Australia<br>3 - Tokyo, Japan<br>4 - Zurich, Switzerland<br>5 - New York, USA<br>6 - Los Angeles, USA<br>7 - Santiago, Chile |

*Output*:

| | | |
|---|---|---|
| lat2 | 32-bit float | latitude in radians |
| long2 | 32-bit float | longitude in radians |
| string buffer | string | name of location<br>e.g. `London, England` |

### getLatLong

Returns the latitude or longitude value from a table of stored values.

*Input*:

| | | |
|---|---|---|
| register 0 | 32-bit integer | table index |

*Output*:

| | | |
|---|---|---|
| register 1 | 32-bit integer | latitude or longitude in radians |

### radiansToDM

Converts a value in radians to a string showing degrees and minutes and stores it at the current string selection point. The format of the string is *DDDD˚MM'*.

*Input*:

| | | |
|---|---|---|
| lat1 | 32-bit float | latitude in radians |
| long1 | 32-bit float | longitude in radians |

*Output*:

| | | |
|---|---|---|
| string selection | string | *DDDD˚MM'*<br>e.g. `-76˚30'` |

### 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 latitude and longitude information. It waits indefinitely until a valid GPRMC sentence is received.

*Input*:

none

*Output*:

| | | |
|---|---|---|
| lat1 | 32-bit float | latitude in radians |
| long1 | 32-bit float | longitude in radians |

### parseGPRMC

This function parses a GPRMC sentence and extracts the latitude and longitude information. Latitude and longitude are converted from NMEA format to radians.

*Input*:

| | |
|---|---|
| string buffer | NMEA sentence |

*Output*:

| | | |
|---|---|---|
| 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 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:

| | |
|---|---|
| *GCdistance.fpu* | contains uM-FPU V3.1 user-defined functions |
| *GCdistance.bs2* | Basic Stamp application to calculate great circle distances |
| *GCdistance.bas* | PICAXE application to calculate great circle distances |

Before running the demo application, the user-defined functions in *GCdistance.fpu* must be programmed into 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*