Carter Coordinate to Latitude and Longitude Conversion Routines

Brandon C. Nuttall
Bnuttall@kgs.mm.uky.edu

KENTUCKY GEOLOGICAL SURVEY
Open File Report 88-08
Revision of June 1990

This distribution consists of the following ten files:

1) CARTERLL.DOC - This file
2) CARTERLL.PAS - The Carter coordinate conversion routines
3) CCLLSAMP.PAS - A sample program
4) FTPERMIN.PAS - A file of typed constants defining width and height in feet on the ground of 1 minute of longitude and 1 minute of latitude by minute of latitude
5) CCLLTEST.DAT - A file of sample data for testing the conversion
6) README.1ST - Last-minute notes and changes; please read
7) CCLLSAMP.EXE - CCLLSAMP.PAS compiled for IBM PC compatibles
8) WELLDATA.DOC - Documentation for using Kentucky Geological Survey well information supplied on diskette
9) KYSAMPLE.DAT - A selection of 103 randomly chosen wells
10) EXAMPLE.PAS - source code of simple example program from documentation

This edition of the documentation revised to reflect the new area code for Lexington, Kentucky. No other changes made.

Note: These conversion routines can be implemented in Microsoft Access or Excel. See the source code for the function _CarterToLatLon() for the algorithm.

Description

This software package provides TURBO Pascal source code and executable routines that perform conversions between Carter coordinate and latitude and longitude locations. A wide range of input formats for the Carter coordinate are supported, and extensive error checking is implemented. The sample program included allows input from the keyboard or from data files, and output may be saved in a file.

These routines are distributed on MS-DOS v3.0 or greater double-sided, double-density diskettes. For systems that use a different operating system, chances are that a utility is available to allow your system to read this format of diskette. Contact Brandon C. Nuttall at the Kentucky Geological Survey, phone 859/257-5500, for information and help with alternate disk formats.

The routines were developed originally on a DEC Rainbow 100B personal computer using the TURBO Pascal compiler, version 3.01a, from Borland International, and have been updated to version 5.0 of the compiler. If you are using another compiler, some changes may have to be made. These changes are discussed later. The functions are implemented as TURBO Pascal units. Additionally, the routines have been run and tested on IBM PC XT- and AT-compatible machines. Please contact Brandon C. Nuttall at the Kentucky Geological Survey, phone 859/257-5500, with questions or problems.

The Carter Coordinate System

The township and range land grid system had not been invented in 1792 when Kentucky became a state. The Carter Oil Company, now EXXON, defined a location grid based on latitude and longitude for the state that is used to locate oil and gas wells. The spacing of the Carter coordinate grid is 5 minutes of longitude by 5 minutes of latitude. This 5-minute Carter section is roughly equivalent to a township. The 5-minute sections are referred to by letters increasing from south to north and numbers increasing from west to east. Each 5-minute Carter section is divided into 25 1-minute sections. The 1-minute sections are about 4,850 feet (east-west) by 6,060 feet (north-south), but this approximation changes with latitude. Locations are specified by supplying: the footages to an adjacent pair of 1-minute section boundaries, designations for which boundaries of the 1-minute section were used, the 1-minute section number, the 5-minute section letter, and the 5-minute section number. By convention at the Kentucky Geological Survey, the 5-minute section letters from "A" to "Z" are preceded by a blank so that the letter range from " A" to "GG" can be sorted into proper order.

The routines in this package perform conversions between the Carter coordinate grid system and latitude and longitude, and account for the change in number of feet per minute on the ground with latitude. The

footages used in the calculations (see FTPERMIN.PAS) are from Special Publication 5 of the U.S. Coast and Geodetic Survey and are to the nearest one-hundredth of a foot. All routines use TURBO Pascal real numbers and therefore maintain 11 or 12 digits during calculation. The Carter coordinate 1-minute footage values calculated from given latitude and longitude values are the footages to the nearest pair of adjacent section boundries and are reported to the nearest foot. For latitude and longitude calculation, results are calculated in degrees and should be reported to no more than six decimal places. One foot on the ground is approximatly 0.0000027 degree of latitude and ranges from 0.0000034 to 0.0000036 degree of longitude; latitude and longitude should be rounded to the nearest 0.000001 degree.

On output, all latitude and longitude values are reported in decimal degrees with six decimal places. However, latitude and longitude values may be input in either decimal degrees or sexagesimal notation. Sexagesimal notation allows the expression of values as conventional degrees, minutes, and seconds. In this notation, a latitude or longitude value is input as three numbers separated by blanks, commas, or by "dms" notation. The number of degrees is a positive or negative integer. The number of minutes is an integer greater than or equal to zero but less than 60. The number of seconds is a decimal number greater than or equal to zero but less than 60.0. In sexagesimal notation, if the number of seconds is zero then the seconds portion may be dropped; if both minutes and seconds are zero then both may be dropped. The following examples illustrate valid input:

1) 37.508333
2) 37.508333d
3) 37d30m30s
4) 37D 30M 30S
5) 37 30 30
6) 37,30,30

Note that if the number of minutes is zero and the number of seconds is greater than zero, no error will be detected and an incorrect value will be calculated, i.e., "37d30s" will be converted to 37.5-degrees not 37.008333-degrees.

Installation

First, use your operating system's copy command to make a copy of the distribution diskette. Put the distribution diskette in a safe place away from your computer. DO NOT USE THE DISTRIBUTION DISKETTE OR MAKE ANY CHANGES TO ANY OF THE FILES ON IT EXCEPT TO MAKE A COPY. For example, on a system with a hard disk and the distribution diskette in drive A:

```
COPY A:*.* C:
```

Using CCLLSAMP.EXE

CCLLSAMP.EXE is an executable version of the program supplied as CCLLSAMP.PAS. CCLLSAMP.PAS demonstrates the usage of the conversion routines. CCLLSAMP.EXE will run on most generic MS-DOS IBM PC-compatible machines. While running CCLLSAMP.EXE, questions that require yes or no answers need only the appropriate "Y" or "N" key. All other entries must be terminated by pressing the enter (or return) key. The program may be terminated at any time by pressing the escape key.

If Carter coordinates or latitude and longitude are input from a file, they are expected to be stored as ASCII strings in a text file (i.e., records are terminated with a carriage-return character). You will be asked about the input data (see the examples that follow). When data are saved to a file, they are saved as ASCII strings written to a text file. Four elements are written to the file, each separated by one blank. The identifier and Carter coordinate strings are supplied as quoted strings, the latitude and longitude as floating-point real numbers. This simple format can be imported into many applications programs including spreadsheets.

To run the sample program, type SAMPLE and press the enter key. An introductory screen will be displayed; press any key to proceed with the example. The first example illustrates calculating latitude and longitude from Carter coordinates using input from the file CCLLTEST.DAT. The dialog and appropriate responses are shown in Table 1.

Table 1. -- Converting Carter Coordinates to Latitude and Longitude Using CCLLTEST.DAT.

```
-----------------------------------------------------------------------
     Do you want to convert latitude and longitude to Carters (y/n)? n
     Do you want to enter input data from the keyboard (y/n)? n
     Enter input data file specification:        cclltest.dat
     Do you want to pause whenever there is an error (y/n)? n
     Enter record length:                        50
     Enter length of record identifier:          7
     Enter offset to start of identifier:        0
     Enter length of Carter coordinate:          20
     Enter offset to start of coordinates:       8
     Do you want to save data to a file (y/n)?   n
-----------------------------------------------------------------------
```

The offset required is the number of characters in the input data set that precede the requested item. The contents of CCLLTEST.DAT are supplied in the appendix. Note that when input is from a file and the results are not being saved to a file, the program will stop and display the results of the calculation regardless of the response to the pause on error query.

Another example, calculating latitude and longitude for the wells supplied in KYSAMPLE.DAT, is shown in Table 2.

Table 2. -- Converting Carter Coordinates to Latitude and Longitude Using KYSAMPLE.DAT.

```
-------------------------------------------------------------------------
     Do you want to convert latitude and longitude to Carters (y/n)? n
     Do you want to enter input data from the keyboard (y/n)? n
     Enter input data file specification:         kysample.dat
     Do you want to pause whenever there is an error (y/n)? n
     Enter record length:                         315
     Enter length of record identifier:           7
     Enter offset to start of identifier:         0
     Enter length of Carter coordinate:           20
     Enter offset to start of coordinates:        96
     Do you want to save data to a file (y/n)?    n
-------------------------------------------------------------------------
```

In this case, the identifier is the Kentucky Geological Survey unique record number. Also, Carter coordinate footages within the 1-minute section are not available for the Anna K. Wilson no. 3 J. S. Roberts well in Metcalfe County; an error message will be generated.

Finally, to illustrate conversion of latitude and longitude to Carter coordinates, an example using CCLLTEST.DAT is shown in Table 3.

Table 3. -- Converting Latitude and Longitude to Carter Coordinates using CCLLTEST.DAT.

```
-------------------------------------------------------------------------
     Do you want to convert latitude and longitude to Carters (y/n)? y
     Do you want to enter input data from the keyboard (y/n)? n
     Do you want the expanded format carter coordinate (y/n)? y
     Enter input data file specification:         cclltest.dat
     Do you want to pause whenever there is an error (y/n)? n
     Enter record length:                         50
     Enter length of record identifier:           7
     Enter offset to start of identifier:         0
     Enter length of latitude:                    10
     Enter offset to start of latitude:           29
     Enter length of longitude:                   10
     Enter offset to start of longitude:          40
     Do you want to save data to a file (y/n)?    n
-------------------------------------------------------------------------
```

Note again, since the Carter coordinates are not being saved to a file, there will be a pause after each error regardless of the response to the appropriate question.

Compiling and Testing

Note, if you wish only to run the sample program, then you do not have to compile anything and may skip this section. To compile the CARTERLL.PAS unit for inclusion into your own programs, a hard drive ("C:") and TURBO Pascal 5.0 will be assumed in the following examples. It is also assumed that the compiler, source files, and all needed units are in the same directory. Examples will be for the command line version of the TURBO Pascal compiler.

Compile CARTERLL.PAS (using "MAKE" option):

```
TPC CARTERLL /M
```

At this point, you are ready to use the compiled CARTERLL unit in your programs by including the statement "USES CARTERLL;" in your source code. (See TURBO Pascal documentation on "USES" syntax.)

To compile the demonstration program:

```
TPC CCLLSAMP
```

CCLLSAMP.EXE is ready to run. See the section on using SAMPLE.EXE to test and run CCLLSAMP. Examine CCLLTEST.DAT and compare your output with the latitude and longitude or Carter coordinates shown. The first 10 entries are tests of error detection and should generate a variety of error messages.

If you incorporate the conversion routines into your own application, you must declare certain variables. See the notes below and examine the source code in CCLLSAMP.PAS.

Function Reference Guide

The Carterll unit provides three functions for performing conversions and validating data. Each of the conversion functions is implemented as two separate functions: a strongly typed "workhorse" function, and a generalized function. The generalized functions provide extensive error checking and formatting and are the primary conversion functions. If you wish to access the workhorse functions, refer to the CARTERLL.PAS source code. Each workhorse function has the same name as the generalized function, and its name is preceded by the underscore character. Finally, each function returns the Boolean value TRUE if and only if the conversion was successful.

CarterToLatLon function (carterll unit)

Function: To convert Carter coordinates to latitude and longitude

Declaration:

```
function CarterToLatLon(Identifier, CarterCoord : string;
                        var Latitude, Longitude : real) : boolean;
```

Remarks: This is a parser for a Carter coordinate string that checks the input and calls _CarterToLatLon() to do the conversion. Sets a pointer to a Boolean variable, CarterFormKnown. If the input Carter coordinate has footages from 1-minute section lines preceding the 1-minute section number and 5-minute section letter and number, then the address pointed to by CarterFormKnown is assigned a value of TRUE. The following are examples of some acceptable formats for Carter coordinate string input:

```
1234 FNL x 2345 FEL 25- G-57
1234FNL2345FEL25 G57
2345FEL1234FNL25 G57
1234FNL2345FEL 25G57
1234FNL2345FEL25 g57
1234N2345E25G57
25G 57 1234n2345e
```

If the 1-minute section number and 5-minute section letter and number precede the footages, there MUST be at least one blank between the 5-minute section number and the first footage.

Example:

```
if CarterToLatLon(id,'1234N2345E25G57',latitude,longitude)
        then writeln(id,latitude,longitude)
        else writeln('Error calculating lat/lon');
```

LatLonToCarter function (carterll unit)

Function: To convert latitude and longitude to Carter coordinates and format the output

Declaration:

```
function LatLonToCarter(Identifier : string;
                        var CarterCoord : string;
                        Latitude, Longitude : real;
                        expand : boolean) : boolean;
```

Remarks: This function converts a latitude and longitude value to a Carter coordinate. If "expand" is set to false, the Carter coordinate string is returned in the compressed form used to store the coordinates at the Survey (i.e., "1234FNL2345FEL25 G57"). Otherwise, the Carter coordinate is formatted for readability (i.e., "1234 FNL x 2345 FEL 25- G-57").

Example:

```
if LatLonToCarter(id,cc,37.013277,-84.824695,true)
        then writeln(id,cc)
        else writeln('Error calculating Carter coordinate');
```

InKentucky function (carterll unit)

Function: To check a letter and number combination for validity

Declaration:

```
function InKentucky(Identifier, SecLetter : string;
                    SecNumber : integer) : boolean;
```

Remarks: This routine provides a consistent implementation and is supplied as a convenience for potential users. The conversion routines access _InKentucky() to do the actual checking.

Example:

```
if InKentucky('TEST','A',57)
        then writeln('Is in Kentucky')
        else writeln('Is not in Kentucky');
```

Using the Routines in Your Application Program

First, the unit name must be included in the "uses" statement. Second, four variables must be declared: one for latitude and one for longitude each of type real, and one for the data identifier and Carter coordinate, both of type string. The data identifier should ideally be unique for each record in the system so that problems can be identified and corrected. Two global constants, "ccllname" and "ccllsource," are available for identifying the program name and the source of the routines. Table 4 shows a short program that illustrates the fundamentals of implementing the conversion routines. The program is provided on the distribution diskette as EXAMPLE.PAS.

Table 4. -- Fundamentals of Using the Conversion Routines.

```
program myprog;
uses
   carterll;                  { include the proper unit }
var
   latitude, longitude : real; { for latitude and longitude }
   cc,                        { carter coordinate string }
   id : string;               { unique identifier }
begin
   writeln(ccllname);         { output name and version of routines }
   writeln(ccllsource);       { output source of routines }
   id := 'TEST';              { set up arbitrary id }
   if CarterToLatLon(id,'1234N2345E25G57',latitude,longitude)
           then writeln(id,latitude:10:6,' ',longitude:10:6)
           else writeln('Error calculating lat/lon');
   if LatLonToCarter(id,cc,37.013277,-84.824695,true)
           then writeln(id,' ',cc)
           else writeln('Error calculating Carter coordinate');
end.
```

Notes

1. The TURBO Pascal type "string" is used for all character strings. If your compiler doesn't have this, use the following declaration:

   ```
   type string = string[255];
   ```

2. This software makes use of TURBO Pascal "units." If your compiler does not support separate compilation and linking of modules, you must either convert CARTERLL.PAS and FTPERMIN.PAS to "include" files or incorporate them directly into your program source code. Contact Brandon C. Nuttall for further details.

3. This software makes use of TURBO Pascal "typed constants." If your compiler does not support this feature, you must convert the typed constants (debug, LatFtPerMin, LonFtPerMin, etc.) to declarations and explicit assignment statements.

4. A boolean variable (really a typed constant) called "debug" is defined and set to false. If the user assigns a value of TRUE to "debug," messages concerning the working of the conversion are printed to the console.

5. Unless an error causes the program to halt, any error detected in the input Carter coordinate causes latitude and longitude to be set to 0.0.

6. The first call to the CarterToLatLon routine causes the input Carter coordinate string to be parsed to determine the format (footages first or section-letter-number first). Once the format is determined, it is assumed to stay that way for every subsequent Carter coordinate. If the format is not constant, the user may elect to force a call to the routine that makes the format determination for each string input by inserting the following lines before each call to CarterToLatLon:

   ```
   Dispose(CarterFormKnown);
   CarterFormKnown := nil;
   ```

   This will cause the routine to forget whatever format determination had previously been made. Note also that the _CCToLatLon routine does not know or care about CarterFormKnown, since its input is strictly typed.

7. By convention at the Kentucky Geological Survey, longitude is returned as a negative number. This reflects the increase of longitude values toward the west. If your program requires positive values, simply multiply the longitude returned by -1.0. If you wish, you may change the source code of function _CCToLatLon() in CARTERLL.PAS. Change the line:

   ```
   Longitude := -1.0*((eedge*1.0)/60.0 + S/3600.0);
   ```

   To:

   ```
   Longitude := (eedge*1.0)/60.0 + S/3600.0;
   ```

   After making this change, you must save and recompile the unit.

APPENDIX

Testing and verification data set, CCLLTEST.DAT

```
_CCER01 9999FNL1450FEL25 M90  "BAD FOOTAGE       "
_CCER02 2250FSL1~00FEL25 M90  "INVALID CHARACTERS"
_CCER03            FEL15 J24  "MISSING PARTS     "
_CCER04 0500FNL0230FEL2   24  "MISSING PARTS     "
_CCER05 1710FNL08.5FWL24 J24  "NOT INTEGER       "
_CCER06 0420FSL2140FNL16 J24  "NOT ADJACENT      "
_CCER07 2363FQL1663FWL17 J24  "Q NOT VALID       "
_CCER08 0850FSL2060FWL33 J24  "1' SEC BAD        "
_CCER09 0520FSL0220FEL16XX24  "5' LTR BAD        "
_CCER10 1650FNL1450FEL25 M99  "5' NUM BAD        "
0039334 1650FNL1450FEL25 M90  37.512135 -82.071665
0039335 2250FSL1600FEL25 M90  37.506179 -82.072182
0023228 3960FNL0430FEL15 J24  37.289124 -87.568145
0025932 0500FNL0230FEL25 J24  37.265293 -87.567457
0025933 1710FNL0825FWL24 J24  37.261970 -87.563832
0038587 0420FSL2140FWL16 J24  37.267820 -87.575980
0047040 2363FSL1663FWL17 J24  37.273156 -87.560952
0052547 0850FSL2060FWL13 J24  37.285668 -87.542920
0052576 0520FSL0220FEL16 J24  37.268095 -87.567423
```

NB: Those data elements with an id beginning with "_CCER" are tests of the ability to detect errors. Processing of each entry should generate an error message and return latitude and longitude values of 0.0. The other entries should return the latitude and longitude values shown in the final columns.