KANSAS GEOLOGICAL SURVEY
OPEN-FILE REPORT 1996-14

PROGRAMMER'S MANUAL FOR LEGXY2: A PROGRAM TO CONVERT LEGAL DESCRIPTIONS IN KANSAS TO UTM ZONE 14 COORDINATES

by

Gregory W. Pouch

Disclaimer

The Kansas Geological Survey does not guarantee this document to be free from errors or inaccuracies and disclaims any responsibility or liability for interpretations based on data used in the production of this document or decisions based thereon. This report is intended to make results of research available at the earliest possible date, but is not intended to constitute final or formal publications.

Kansas Geological Survey
1930 Constant Avenue
University of Kansas
Lawrence, KS 66047-3726

Programmer's Manual for LEGXY2: A Program to Convert Legal Descriptions in Kansas to UTM Zone 14 Coordinates

Gregory W. Pouch
Geohydrology Section
Kansas Geological Survey

Abstract

LEGXY2 is a Microsoft Windows program that allows the user to convert Kansas legal (USPLSS) point descriptions to UTM Zone 14 Coordinates. USPLSS legal descriptions can be entered interactively or be obtained from legal descriptions stored in the tables or queries of a Microsoft Access database. LEGXY2 relies upon a database to provide the section coordinates, then further subdivides the section using simple linear interpolation or parallel offset. Both center-of-subdivision and footage point locations can be used.

LEGXY2 was written in Visual Basic 3.0 Professional Edition, and uses the database access features. With minor effort it could be adapted to Microsoft Access Basic, and with major effort to Microsoft Excel's Visual Basic for Applications. The interpolation scheme outlined here corresponds closely to surveying practice for regularly-shaped sections. For footages, LEGXY2 offsets the corners parallel to section lines. For center-of-subdivision descriptions, LEGXY2 calculates fractional sections east and north of the southwest corner and uses a weighted average method to find the intersection. LEGXY2 is limited by the availability of section information, and uses only a four-point section outline.

Table of Contents

Abstract 1
Table of Contents 2
List of Figures 2
Introduction 3
Purpose of program 3
Overview of Program Design 3
Disclaimer 4
Acknowledgments 5
Related Software and Documents 5
The US Public Land Survey System (USPLSS) 5
Purpose of the US Public Land Survey System 5
Nomenclature 6
Implementation of the USPLSS 9
Limitations of LEGXY2 9
Meanders 10
Oddly-shaped sections 10
No use of center-of-edge points 10
Coordinate Conversion Algorithms 10
Finding Sections Using KSSAS 11
Footages 13
Center-of-Subdivision 14
Convert to Fraction 14
Interpolate 15
Recommendations for Surveying for Electronic Mapping 15
References 15
Appendix 1: Complete Code Listing on Following Pages 16

List of Figures

Figure 1 Idealized diagram of the USPLSS: tiers, ranges, townships and sections 8
Figure 2 Idealized township showing the KSSAS column and row offsets 12
Figure 3 Diagram of a section for footages 13
Figure 4 Schematic section for center-of-subdivision with monumented locations 14

Introduction

PURPOSE OF PROGRAM

LEGXY002 converts point locations described in terms of "**leg**al" coordinates (given relative to the Public Land Survey System) to standard **X**, **Y** coordinates for navigation and for use with spatial analysis and display software. This program relies on a database containing section outlines for the state of Kansas, descended from Ross's (1994) earlier LeoBase II database, along with serial numbers assigned according to the KSSAS scheme described herein.

The United States Public Land Survey System (USPLSS) provides the legal basis for land ownership in most of the United States west of the Appalachians. Under this system, an area was to be divided into square townships six miles on a side and oriented along cardinal directions, then subdivided into sections a mile square and oriented along cardinal directions. (See Figure 1.) Because property lines are based on section lines and field boundaries are often readily apparent in the field, point locations are often specified as being at/near the center of some subdivision of a section, or so many feet east and north of a section corner. This is particularly true of oil, gas, and water wells. The advantage of specifying a point location in terms of the USPLSS is that the location also implicitly names ownership.

Because the USPLSS is a land ownership system and not a grid system, once an area was surveyed, the section corners are not moved. A section is the area enclosed by a set of survey markers laid out by the original survey, and is not subject to readjustment to correct errors made during the survey. Thus, converting from legal descriptions to a physical coordinate system such as longitude-latitude or UTM requires knowledge of the locations of the section corners. LEGXY2 uses the sections in Kansas digitized from 7 ½' quadrangles (Ross, 1994), although updated section coordinates could be easily used.

Producing maps with accurate locations requires use of a more standardized, physical coordinate system, such as longitude-latitude, or distances east and north of some arbitrary point, such as the Universal Transverse Mercator (UTM) system. LEGXY2, the program described herein, converts LEGal descriptions to X,Y coordinates in the UTM Zone 14 coordinate system for all of Kansas.

OVERVIEW OF PROGRAM DESIGN

From a programming perspective, there are three main tasks in LEGXY2 1) get the section and footages or subdivisions from the user or database [input], 2) calculate the resulting X,Y locations [calculate],and 3) display or store the results [output]. The code is lavishly commented and variables usually have self-documenting names.

Because the input and output steps depend on the operating system, programming environment, and are very difficult to explain without executing the code in single-step mode, the reader is advised to examine the user-interface code directly and single-step through some example sessions: this manual does not explain the internal workings of the user interface.

The calculation of coordinates does not depend on the particulars of the computer and is described below and in a later section. For point locations provided as offsets from a corner, also commonly known as footages, the program calculates unit vectors parallel to the section lines that intersect at that corner and offsets the corner coordinates appropriately. For point locations described as "center of the NW¼ or the SW¼", a more complicated scheme involving first parsing the description to fractional sections then linearly-interpolating is used.

DISCLAIMER

The locations of section corners used by LEGXY2 are approximate, and the interpolation scheme is not the one used in defining property boundaries. The user should not use LEGXY2 in lieu of property or boundary surveys, or in any other cases where a 200 foot error would be unacceptable. This program is intended only for use in geology where approximate locations are considered adequate. Use at your own risk.

The database structure, database software, and processing software described in this document are the work of the Kansas Geological Survey, University of Kansas. The Kansas Geological Survey, University of Kansas, makes no warranty or representation, either express or implied, with respect to the database, documentation or interpretations or decisions based on data processed using this software, including their quality, performance, merchantability, or fitness for a particular purpose. In no event will the Kansas Geological Survey, University of Kansas, be liable for direct, indirect, special, incidental, punitive, or consequential damages arising out of the use of or inability to use the database or documentation whether based upon contract, negligence, strict liability or otherwise.

The Kansas Geological Survey, University of Kansas, does not and will not guarantee the accuracy of this data. This product is based on provisional data made available solely for the convenience of interested researchers with the understanding that it is provisional, unchecked and that the user will not hold the KGS liable in any way for the accuracy or completeness of this data.

The data in this database is provisional data and unchecked. The data in this database may, and almost certainly does, contain errors, both in the original data and in the process of importing it into this database. The Kansas Geological Survey, University of Kansas, does not warrant that the database will meet your requirements, that it will operate in an error free manner, or that it contains accurate and complete data.

If you use this software and database, you are accepting the terms of this disclaimer (i.e., use at your own risk.). If you do not or cannot agree to these terms, return the software and database immediately.

UNDER NO CIRCUMSTANCES SHOULD YOU RE-DISTRIBUTE THIS DATA OR THIS SOFTWARE APART FROM THIS DISCLAIMER.

ACKNOWLEDGMENTS

The author wishes to acknowledge the work Charles Ross who wrote LeoBase, a Legal to Longitude, Latitude conversion program, and all those who digitized the section corners originally; and Jim Mitchell, who provided the author with a very useful wrapper program for use with Arc/Info that produced GENERATE format files of section outlines, which could then be imported using WHEAT.

RELATED SOFTWARE AND DOCUMENTS

The section lines used in this database are descended from those developed by KGS by digitizing the sections in Kansas from 7 ½' quadrangles (Ross, 1994). They are stored in WHEAT XYChain format (see Pouch, 1996a). The user's manual for this program is available as a KGS Open File Report (Pouch, 1996b).

A more comprehensive discussion of the US Public Land Survey System can be found in most surveying textbooks, such as Brinker and Wolf (1984)

The US Public Land Survey System (USPLSS)

This discussion substantially repeats that of Brinker and Wolf (1984). Similar explanations of the USPLSS can be found in almost all American surveying textbooks and many soil science and geologic field mapping texts. If you have already taken a surveying course, there is no need to read this. If you think all sections are regularly-arranged squares 5280 feet on a side, make every effort forget that idea entirely and read this.

PURPOSE OF THE US PUBLIC LAND SURVEY SYSTEM

During colonization of the United States, much land was originally owned by the federal government and later sold, starting with the Northwest Territories (Great Lakes region). The Continental Congress, many of whom were surveyors, developed a rationalized system of subdividing public lands for ownership known as the Public Land Survey. Because the USPLSS is an ownership system and not a navigation system, boundaries are fixed once certified by the authorized surveyor. It is what it is, not what it should have been.

Traditionally, land descriptions had been on the metes and bounds system: "metes" are measured distances and "bounds" refers to boundary lines. A property description in metes and bounds includes a starting point (a fence, lake, stake, some other property...) and a series of directions for traversing each point on the boundary. Ideally, these are distances (in whatever units the original survey was in) and bearings (referenced to astronomical, magnetic, or assumed north) to each to point on the boundary, but they are sometimes much more difficult to interpret, let alone re-occupy. Sometimes, the deed has a starting point of the form "an apple tree five minutes walk from..."; often, the landmarks can no longer be found or have been moved or obliterated, and so forth.

Because of the difficulty of re-establishing boundaries based on the metes and bounds system, the Continental Congress adopted a grid-based system. The goals were to produce sections with boundary lines oriented along cardinal directions, one mile on a

side, and six-hundred forty acres in area: these conditions cannot all be met on a round earth.

To try meeting all these conditions and accommodate the shape of the earth, a more elaborate scheme involving occasional offsets, auxiliary baselines, secondary meridians, and such was developed as a practical solution. This is described below, mainly to explain why large offsets occur in certain areas but not others. The main points are 1) that a principal goal of the USPLSS was to produce as many equal-sized sections as possible, so errors were propagated into the north and west edges of townships and 2) USPLSS is a land ownership system, not a grid system or navigational system.

NOMENCLATURE

The USPLSS is a grid-based system for describing land ownership. The goal was to divide the land into hierarchly arranged sections. At the top of this hierarchy are Principal Meridians, which are arbitrary starting points for the original surveys. All of Kansas was surveyed on the Sixth Principal Meridian (6th P.M.). (Principal Meridians typically encompass areas of the order of hundreds of miles.)

More familiar to most geologists are next three levels of this hierarchy: tiers, ranges, sections, and townships. ("Township" is used in a somewhat restricted sense to refer only to a "six"-mile square.)

A **tier** is an east-west swath of land intended to be six miles in north-south extent: tiers are numbered from the baseline with an indicator of north or south (N or S). The Kansas Geological Survey is located in the thirteenth tier south of the Baseline, written as T13S or 13S.

A **range** is a north-south swath of land intended to be six miles in east-west extent: ranges are numbered from the meridian with an indicator of east or west (E or W). KGS is in the nineteenth range east of the Meridian, written as R19E or 19E.

A **township** is the intersection of a tier and a range. KGS is therefore in T13S, R19E. There are also political townships with which have nothing to do with the USPLSS.

A **section** is parcel of land, which was to intended to be cardinally-oriented, one mile on a side, and having an area of 640 acres. KGS is located in section 2 of T13S R19E, written as T13S, R19E, S 2 or 13S19E02 or S 2 of R19E, T13S.

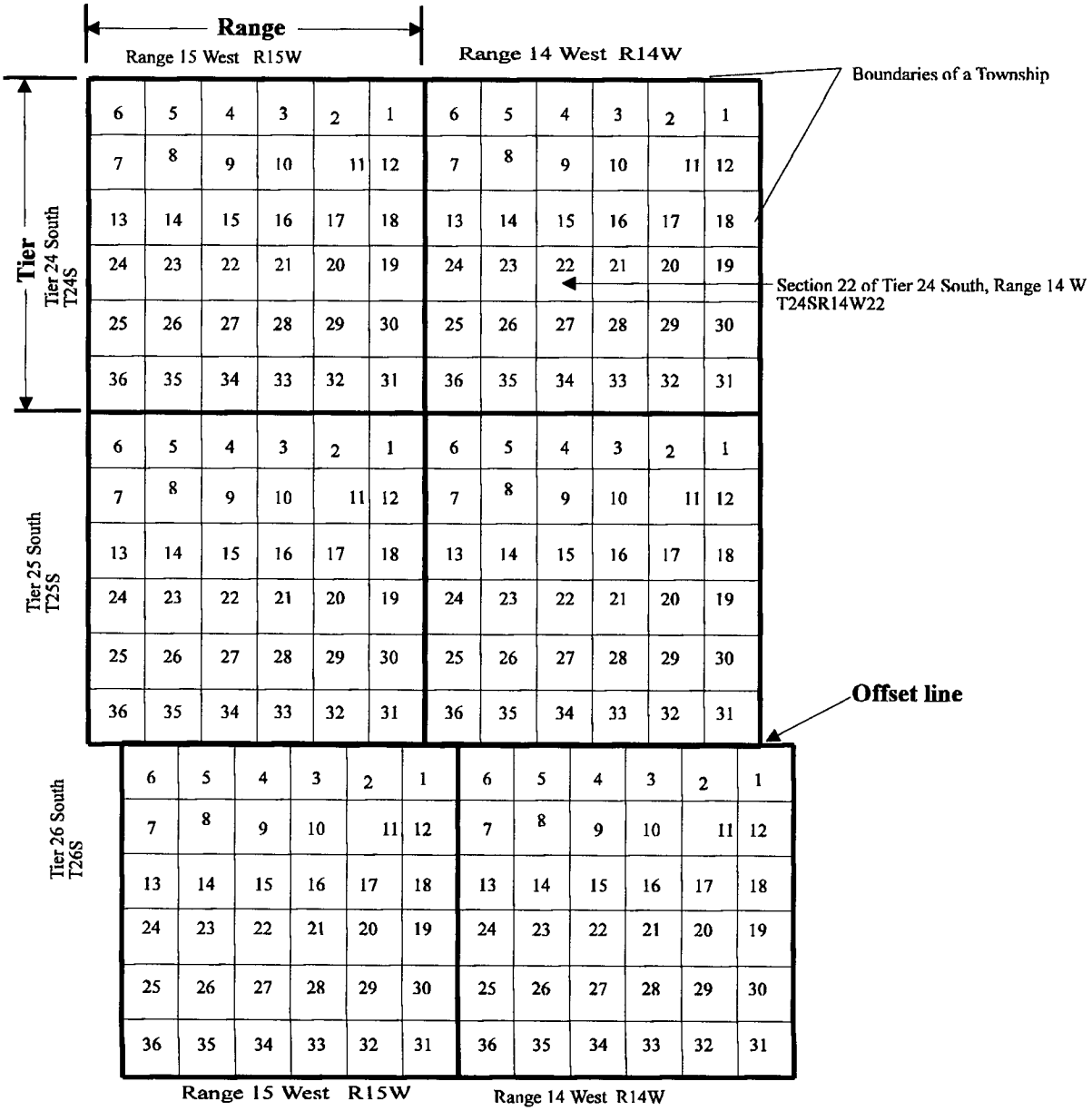
A section is commonly believed to be the basic unit of the USPLSS, but it is not. Instead, a **quarter-section**, a land parcel a "half"-mile in extent, is the smallest unit measured in the USPLSS: the "half"-mile marks along survey lines define most of a quarter section. The center of the section is defined implicitly as the intersection of two lines, one connecting the half-way marks on the east and west edges, and the other connecting the half-way marks on the north and south edges. KGS is located in the southeast quarter of its section, so it would be in 13S19E02 SE¼.

To describe smaller subdivisions and points, two main systems are used: quarterings and footages. **Footages** describe point locations from a section corner in feet east/west along the southern/northern section line and north/south along the eastern/western section line. In **quarterings**, the halving system begun in the original survey is continued: a point location might be described as being in the NE¼ of the NE¼ of the SE¼ of its section. (Guess where KGS is.) Unfortunately, it might also be

described as the SE¼ NE¼ NE¼. Usually, the first ordering is used, and a good rule of thumb is to assume that descriptions always proceed either from largest to smallest or from smallest to largest, so the order of section and township gives a useful hint. The US Bureau of Land Management promulgated a system to overcome this ambiguity, in which the quarters are lettered A for NE, B for NW, C for SW and D for SE. These ALWAYS proceed from largest to smallest, so the KGS is in 13S19E02 DAA. Ideally, the first quartering is based on the original delineation of half-mile quarter-sections and subsequent subdivisions are based on interpolation, but in common practice, followed in LEGXY2, all subdivisions are based on interpolation.

Figure 1 Idealized diagram of the USPLSS showing tiers, ranges, townships and sections

The small blocks are sections: they are all drawn the same size for convenience only. Sections come in groups of 36 in a township and are numbered as shown. A township is the intersection of a tier and a range. A tier runs east-west and gives north-south (Y) position: a tier is often called a township as well. A range runs north-south and gives east-west (X) position. Irregularities are particularly common in the northern row and western column of a township, along offset lines, and near bodies of water, but occur throughout.



IMPLEMENTATION OF THE USPLSS

This section is included to give the user some idea why the USPLSS is not a navigational grid. It is not necessary to understanding how the program works.

The first step in surveying an area under the public lands system was to choose an initial point then survey a north-south meridian, setting monuments every ½ mile. A baseline, perpendicular to the meridian at the initial point and running due east-west, was surveyed, again setting monuments every ½ mile. A principal meridian and its baseline form the basis for the rest of the system, and an area several hundred miles across was usually surveyed on a single meridian. All of Kansas is surveyed on the Sixth Principal Meridian.

After establishing the principal meridian and baseline, a set of standard parallels (parallel to the baseline and offset by 24, 30 or 36 miles) were then run, followed by a set guide meridians (parallel to the principal meridian and offset by 24, 30 or 36 miles): this resulted in a set of quadrangles 24, 30, or 36 miles square, again setting markers every ½ mile. Next, the townships were marked off by connecting the appropriate markers along the guide meridians and standard parallels. Finally, sections and quarter-sections were marked off along those lines by setting markers every ½ mile. Any error introduced affected all subsequent steps.

The goal was to produce a set of sections, cardinally-oriented, with markers at the NW, NE, SE, and SW corners and at the "midpoints" of the section edges (e.g., there are also markers at the middle of the north, east, south, and west section lines). The center of the section is implicitly defined as the intersection of the line connecting the mid-point markers on the east and west edges with the line connecting the mid-point markers on the north and south edges. The actual entities surveyed are quarter-sections, bounded by the "section" corners, the "mid-point" markers, and the implicit center-point.

One of the principal goals was to create as many equal-sized sections as possible, so closure errors were propagated into particular areas: the north and west parts of townships. Another goal was to reduce unproductive travel by the field party. Because of this, the USPLSS has very irregularly distributed errors.

The foregoing describes the goal, not the result. Remember, USPLSS is a land ownership system, not a navigational grid. Before criticizing irregularities in the USPLSS surveys, remember that these were done with primitive equipment, far from home and under primitive conditions, even for the time.

Limitations of LEGXY2

The preceding discussion helps explain some of the limitations built into LEGXY2. There are aspects of the USPLSS that were not included, due to lack of available data and the difficulty of incorporating these into a computer program. For these situations, it is probably better to get the quadrangle maps, post the points, and pull the coordinates manually.

MEANDERS

The preceding discussion described how sections are arranged in open areas, free of obstructing water bodies. In such areas, there are additional corners, called meander corners, that complicate this whole process. LEGXY2 makes no attempt to deal with meanders.

ODDLY-SHAPED SECTIONS

As stated earlier, there are actually eight surveyed and one-implicit staked locations in a section. A section is actually an octagon, although the goal was to make four sides, each with three co-linear points. Because all data available to me used the four-corner representation of sections, no attempt was made to incorporate the more accurate eight-corner definitions.

Along the south and west edges of the state, there are a number of under-sized sections, where the survey boundary is truncated by the state line. Depending on your viewpoint, there are sections that only have eastern quarters (probably the better interpretation) or whole sections with odd shapes (the method used in LEGXY2). The query used to retrieve section coordinates from the database could be modified by the end-user to not return coordinates in such cases.

NO USE OF CENTER-OF-EDGE POINTS

As stated earlier, the first-level quartering of a section should be based on the staked locations and subsequent quarterings on interpolation. LEGXY2, like most other software for this purpose, uses only the four main section corners for the whole process. Were eight-point data to become available, LEGXY2 could be modified to take advantage of this.

Coordinate Conversion Algorithms

Calculating coordinates from a legal description involves the following steps

1) Get the section outline [GetSectionCoords( ) in LEGXY013.FRM]
2) Identify the section corners (i.e., figure out which point on the perimeter is the NE corner) [GetCornersOfSection( ) in LEGXY003.BAS]

3A) For footages [XYFrom_OutlineCornrOffst( ) in LEGXY003.BAS]
- 3A1) Find unit vectors along the boundaries that meet at the specified corner
- 3A2) Multiply the footages by the unit vectors
- 3A3) Add the X,Y offset to the corner location

3B) For subdivisions
- 3B1) Parse the subdivision to specify fractional section north and east [{dXFromABCD( ) and dYFromABCD( ) } or {dXFromSubsectSEofNW( ) and dYFromSubsectSEofNW( ) } in LEGXY003.BAS]
- 3B2) Find the location [XYFromDXDYQuad( ) in LEGXY003.BAS]

FINDING SECTIONS USING KSSAS

Quickly calculating locations based on section descriptions requires quickly locating the section coordinates. LEGXY2 uses a pre-compiled parameter query and a section serial number, explained below, to retrieve the section boundary. If you are not interested in the section addressing system, there is no need to read this section.

Using the section name to retrieve the section outlines is slow and error-prone, especially since the user could alter the section name by dropping or adding leading zeroes, remove the E/W because they only work west of the principal meridian, etc. To avoid these potential errors and produce a single 4-byte serial number that can be used for fast retrieval, the Kansas Statewide Section Addressing System (KSSAS) was developed. KSSAS assigns two two-byte integer addresses to each section, one indicating its row, the other its column. KSSAS_Row always increases to the north, KSSAS_Column always increases to the east. (The Row, Column representation also facilitates quick estimation of distances and adjacency, because they increase monotonically with direction, unlike section numbers. KSSAS is a lot like the 100 South, 200 East designation of county roads, but is organized to work on a statewide basis.) The section serial number is the four-byte binary concatenation of these two two-byte integers.

Offsets are provided so that Rows are always four-digit numbers and Columns are always five-digit numbers within Kansas. The step size from one section to the next is 32 so that subsections to the fifth-level quartering (0.625 acre plot) can also be given integral row and column designations, although this feature is not currently used.

The KSSAS address is calculated using the functions and constants in the module KSSAS004.bas. These functions rely on integer division and remainders to produce the section addresses. To produce the section serial number from the section column and row, the two two-byte integers are placed into a structure (TYPINTXY), copied to another structure (TYPLONG) using the LSet operator, and extracted [See kssas_SerialNumberFromTTdRRdSS( ) in KSSAS004.BAS]

The KSSAS004.BAS module includes both forward and reverse transformations. Figure 2 shows KSSAS Row and Column offsets for a township. KSSAS_Column for a section is calculated by adding the section's column offset to the base column for the range of the township. KSSAS_Row is calculated by adding the section's row offset to the base row for the tier of the township. Subsections would add their own offsets, but this feature is not used.

Figure 2 Idealized township showing the KSSAS column and row offsets for each section.

6 (0, 160)	**5** (32, 160)	**4** (64, 160)	**3** (96, 160)	**2** (128, 160)	**1** (160, 160)
7 (0, 128)	**8** (32, 128)	**9** (64, 128)	**10** (96, 128)	**11** (128, 128)	**12** (160, 128)
18 (0, 96)	**17** (32, 96)	**16** (64, 96)	**15** (96, 96)	**14** (128, 96)	**13** (160, 96)
19 (0, 64)	**20** (32, 64)	**21** (64, 64)	**22** (96, 64)	**23** (128, 64)	**24** (160, 64)
30 (0, 32)	**29** (32, 32)	**28** (64, 32)	**27** (96, 32)	**26** (128, 32)	**25** (160, 32)
31 (0, 0)	**32** (32, 0)	**33** (64, 0)	**34** (96, 0)	**35** (128, 0)	**36** (160, 0)

FOOTAGES

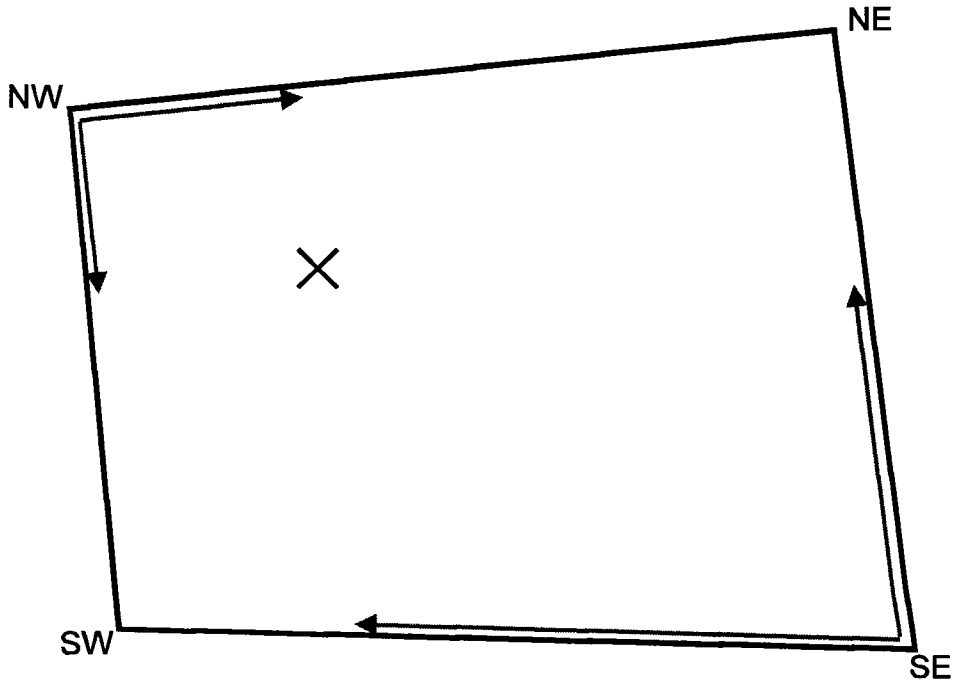
Footages are point locations specified in terms of a section corner and offsets along the section lines that meet at that corner. See Figure 3.

Determining coordinates for a point location described by footages is surprisingly easy. Get the section coordinates. Find unit vectors along the section lines that meet at the corner of interest (e.g., if footages are from the NE corner, find a unit vector from the NE corner to the NW corner for the X-offset, and a unit vector from the NE corner to the SE corner for the Y-offset), multiply these unit vectors by their corresponding offsets, and add both to the corner location to get the point location. See XYFrom_OutlineCornrOffst( ) in LEGXY003.BAS.

This process is exactly invertible: given the X,Y location and the corner of reference, it is possible to find exactly the offsets along the section lines. It is also possible to choose the nearest corner for the reference corner. See OffsetCornrFromXY_Outln ( ) in LEGXY003.BAS.

Figure 3 Diagram of a section for footages

The X can be referenced with footages from any of the section corners. This diagram shows the NW and SE corners. Distances are measured along section lines, not astronomical north. Due to the irregular shape of sections, the footage east from the northwest corner and footage west from the southeast corner will not total 5280 feet.

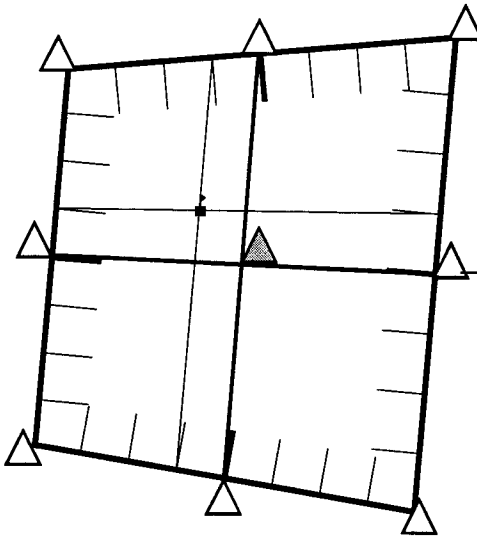


CENTER-OF-SUBDIVISION

Point locations can be described as being at/near the center of some subdivision or being in some subdivision, such as the SE¼ NE¼ NE¼ of S 01 T13S R19E. This is not really a good way to describe point locations, but it is a common way to describe point locations. LEGXY2 converts point center-of-subdivision descriptions to X,Y locations by a two-step process: convert the description to two fractions, one for X and one for Y, and perform the interpolation. Note that the first quartering should be determined by selecting a quarter-section based on an eight-point section outline, followed by interpolation: no eight-point data are presently available, so only the four-point interpolation scheme is used.

Figure 4 Schematic section for center-of-subdivision with monumented locations

This figure shows a school drawn in the SE¼ of the NW¼ of the section. On the east and west edges, the points 5/8 of the distance from south to north are plotted and connected. On the north and south edges, the points 3/8 of the distance from west to east are connected. The intersection is the center of the SE¼ of the NW¼



Monumented locations from the US Public Land Survey System. The center point is implicitly located by connecting the quarter-section monuments shown.

Convert to Fraction

Subdivision locations are commonly expressed as NW¼ of the SE¼ or using the Bureau of Land Management's ABDC lettering system. The latter is preferred, due to its standardization. The former is usually written smallest to largest, but sometimes smallest to largest. LEGXY2 can convert subdivision on the smallest to largest NW¼ of SE¼ system or on the BLM's ABDC system.

To find fractional sections, start at 0.5 (the middle). Proceed from largest quartering to smallest. For each successive quartering, add/subtract a power of ½ based on the position within the string and whether the letter is W/E or N/S. See functions like dXFromSubsectSEofNW( ) in LEGXY003.BAS for examples

Interpolate

The following formulas are used to calculate the X,Y of a center-of-section point, based on fractional section and the corner locations extracted from the section outline. This is used in XYFromDXDYQuad( ) to calculate the X,Y location.

```
XOut = (1 - dY) * ((1 - dX) * SWCorner.X + (dX) * SECorner.X) + (dY) * ((1 - dX) *
    NWCorner.X + (dX) * NECorner.X)
YOut = (1 - dX) * ((1 - dY) * SWCorner.Y + (dY) * NWCorner.Y) + (dX) * ((1 - dY) *
    SECorner.Y + (dY) * NECorner.Y)
```

Because there are cross-terms in dX and dY (fractional sections), it is impossible to invert this to determine dX,dY from X,Y. The best that can be accomplished is to find all centers to a given level of subdivision and find the closest.

Recommendations for Surveying for Electronic Mapping

Computer-based mapping and spatial analysis requires feature coordinates in a an absolute, regular, metric coordinate system, such as UTM or State Plane Coordinates.

The United States Public Land Survey System is a land ownership system. Point locations are often stated relative to the USPLSS. LEGXY2 converts from LEGal descriptions to X,Y locations as accurately as possible, given the constraints of available data.

Given the availability of cheap and easy to use Global Positioning System (GPS) receivers, new locations should be surveyed in ("GPSed in"). If old locations are re-visited, they should also be GPSed in the course of fieldwork and these coordinates used instead of the legal description.

LEGXY2 allows the user to convert from Legal to X,Y coordinates, given the limitations of the available data and the methodology used. If there is need to provide both legal and X,Y descriptions of a point location, footages (corner and offsets) should be used instead of the center-of-subdivision system.

References

Brinker, Russell C., and Paul R. Wolf, 1984, *Elementary Surveying*, Harper & Row, 608 pages.

Pouch, Gregory W., 1996a, *WHEAT Software Development Toolkit*, Open File Report 96-12, Kansas Geological Survey.

Pouch, Gregory W., 1996b, *LEGXY2 A Microsoft Windows Program to Convert Legal Descriptions in Kansas to UTM Zone 14 Coordinates*, Open File Report 96-13, Kansas Geological Survey.

Ross, Charles G., 1994, *Leo II Automated Conversion between Legal and Geographic Reference for Locations in Kansas*, Open File Report 93-53, Kansas Geological Survey.

Appendix 1: Complete Code Listing on Following Pages

```
frmLegalToXY_DB_BOTH - 1

Option Explicit
          Dim a As String
Dim DBName As String ' May want to change scope of this later.
Dim intCoordinateField As Long
Dim XYBlob As String
Dim DataPt() As RealPoint

Dim TabChar As String * 1, CR  As String * 1, CRLF  As String * 2

Dim nFields As Long, n As Long

Dim DB As Database, tblLineData As table
Dim snpQTList As Snapshot, dnsCurrentQT As Dynaset


Dim SepName(0 To 20) As String, SepChar(0 To 20) As String
Const X_INDEX = 0
Const Y_INDEX = 1
Dim SectionDBName As String, strBLOBField As String
Dim SectionsDB As Database, snpSmall As Snapshot
Dim sngSectionOutline() As Single


'For the fast, parameter query method
Dim ParamQueryDef As QueryDef, strParamQueryName As String
'For the slow, table hunt method
Dim snpLarge As Snapshot
Dim strSectTable As String, strSerialNumberField As String

'For "footages"
Dim CornerFrom As String * 2, UnitMultiplier As Single


Dim flgIHaveCurrentSection As Long, DummyLong  As Long, flgUseParameterQuery As Long, strTemp A
s String
Dim NECorner As RealPoint, NWCorner As RealPoint, SWCorner As RealPoint, SECorner As RealPoint,
 BadCenter As RealPoint


Dim ColOffset  As Single, RowOffset As Single
Dim XOut As Single, YOut As Single

Dim strCurrentSection As String, strCurSubdivision As String, lngSectionSerialNumber As Long
Dim intColForSection As Integer, intRowForSection As Integer

Dim MouseX As Single, MouseY As Single

Dim flgIgnoreChanges As Long, flgOffsetsFrom As Long
'Values for use with   flgOffsetsFrom
    Const OFFSET_FROM_FOOTAGE = 1
    Const OFFSET_FROM_NWofNE = 2
    Const OFFSET_FROM_ABCD = 4
    Const OFFSET_FROM_MAPOID = 8

Const WHITE = 16777215




' Shift parameter masks
 Const SHIFT_MASK = 1
 Const CTRL_MASK = 2
 Const ALT_MASK = 4

' Button parameter masks
 Const LEFT_BUTTON = 1
 Const RIGHT_BUTTON = 2
```

```
frmLegalToXY_DB_BOTH - 2

 Const MIDDLE_BUTTON = 4

' Function Parameters
' MsgBox parameters
 Const MB_OK = 0                  ' OK button only
 Const MB_OKCANCEL = 1            ' OK and Cancel buttons
 Const MB_ABORTRETRYIGNORE = 2    ' Abort, Retry, and Ignore buttons
 Const MB_YESNOCANCEL = 3         ' Yes, No, and Cancel buttons
 Const MB_YESNO = 4               ' Yes and No buttons
 Const MB_RETRYCANCEL = 5         ' Retry and Cancel buttons

 Const MB_ICONSTOP = 16           ' Critical message
 Const MB_ICONQUESTION = 32       ' Warning query
 Const MB_ICONEXCLAMATION = 48    ' Warning message
 Const MB_ICONINFORMATION = 64    ' Information message

 Const MB_APPLMODAL = 0           ' Application Modal Message Box
 Const MB_DEFBUTTON1 = 0          ' First button is default
 Const MB_DEFBUTTON2 = 256        ' Second button is default
 Const MB_DEFBUTTON3 = 512        ' Third button is default
 Const MB_SYSTEMMODAL = 4096      'System Modal

' MsgBox return values
 Const IDOK = 1                   ' OK button pressed
 Const IDCANCEL = 2               ' Cancel button pressed
 Const IDABORT = 3                ' Abort button pressed
 Const IDRETRY = 4                ' Retry button pressed
 Const IDIGNORE = 5               ' Ignore button pressed
 Const IDYES = 6                  ' Yes button pressed
 Const IDNO = 7                   ' No button pressed

Const DOUBLEQUOTE = """"

Dim flgDataEntryStyle As Long
Const DATA_ENTRY_MANUAL = 0
Const DATA_ENTRY_BATCH = 1

Private Sub AddFieldsForOutput()
Dim Temp As String

Dim name_SectionsFrom As String, name_XTo As String, name_YTo As String
    name_SectionsFrom = cmbSectionName.Text

    name_XTo = Trim$(txtXFieldOut.Text): If name_XTo = "" Then name_XTo = "X_FromLegalDescripti
on"
    name_YTo = Trim$(txtYFieldOut.Text): If name_YTo = "" Then name_YTo = "Y_FromLegalDescripti
on"

On Error Resume Next
'Figure out whether the X and Y OUT fields exist
   Temp = dat_SectionConversion.Recordset.Fields(name_XTo)
   Temp = dat_SectionConversion.Recordset.Fields(name_YTo)

   If Err <> 0 Then  'For now, trap all errors by creating the fields
     'Figure out where the fields should go
     Err = 0
    Dim strSourceTable As String
    strSourceTable = dat_SectionConversion.Recordset.Fields(name_SectionsFrom).SourceTable

     'Create both field
        Dim nfldX_Long As New field, nfldY_Lat As New field
        nfldX_Long.Name = name_XTo: nfldX_Long.Type = DB_SINGLE
        nfldY_Lat.Name = name_YTo: nfldY_Lat.Type = DB_SINGLE

        Temp = dat_SectionConversion.RecordSource
        dat_SectionConversion.RecordSource = "": dat_SectionConversion.Refresh
        'Professional Edition of VB 3.0 only allows this addition of a field
        DB.TableDefs(strSourceTable).Fields.Append nfldX_Long
        DB.TableDefs(strSourceTable).Fields.Append nfldY_Lat
```

```
        dat_SectionConversion.RecordSource = Temp: dat_SectionConversion.Refresh
    End If

End Sub

Private Function CheckQTInputs() As Long
'Make sure we have a section field
If cmbSectionName.ListIndex < 0 Then
    MsgBox "Your query or table must indicate a section, and the list of sections does not cont
ain one.  Please enter a field name in the list box labelled Field containing section name as T
TdRRdSS."
    CheckQTInputs = False
    Exit Function
End If
If InStr(CStr(cmbSectionName.Text), """") Then
    MsgBox "Your query or table must indicate a section, and the list of sections does not cont
ain one.  Please enter a field name in the list box labelled Field containing section name as T
TdRRdSS."
    CheckQTInputs = False
    Exit Function
End If

'  If the user has not chosen  either the
'  subdivision or footage offsets, ask if they want to
' use the section centers.
Dim iTemp As Integer


If optFootage = False And optSubSection = False Then
    iTemp = MsgBox("You have not chosen subsections or footages as an option.  Did you want jus
t the section centers?", MB_YESNOCANCEL Or MB_ICONQUESTION Or MB_DEFBUTTON3)
    Select Case iTemp
        Case IDYES
           Call optSubSection_Click(True)

           optAsABCD = True
           cmbABCD.AddItem """__"""
           cmbABCD.ListIndex = cmbABCD.ListCount - 1
        Case IDNO, IDCANCEL
           MsgBox "Choose one of the option buttons for FOOTAGES or QUARTERING, choose fields o
r constants within the frame beneath it, then try the conversion again."
           CheckQTInputs = False
           Exit Function
    End Select
End If

If optSubSection And optAsNWofNE = False And optAsABCD = False Then
    MsgBox "You have chosen to use Subdivision of sections as a location method, but have not c
hosen between NW1/4 of NE1/4 format and ABCD format.  Please do so then try the conversion agai
n."
    CheckQTInputs = False
    Exit Function
End If


CheckQTInputs = True


End Function

Private Sub ClearFootages()
''Clear the entries in the footage locations
  flgIgnoreChanges = True
  zlblOffset = "No Corner Selected"
  txtEWOffset = "": txtEWOffset.BackColor = WHITE
  txtNSOffset = "": txtNSOffset.BackColor = WHITE
  CornerFrom = "  "
  txtX = "": txtY = ""
   optNW = False: optNE = False: optSW = False: optSE = False
```

```
  flgIgnoreChanges = False
End Sub

Private Sub ClearSubSections()
  flgIgnoreChanges = True
    txtSubNWNE = "": txtSubNWNE.BackColor = WHITE
    txtSubsectionABD = "": txtSubsectionABD.BackColor = WHITE
    txtX = "": txtY = ""
  flgIgnoreChanges = False
End Sub

Private Sub cmbAllFields_KeyDown(KeyCode As Integer, Shift As Integer)
     Debug.Print KeyCode, Shift
End Sub

Private Sub cmbQTList_SubSection_Click()
    Dim Temp As String
    Temp = cmbQTList_SubSection
    Call SetQuery_Sections(Temp)


    optFootage = False
    optSubSection = False
    optAsNWofNE = False
    optAsABCD = False

      'DISable footage related controls
        fraFootage.Enabled = False
        cmbFromCorner.Enabled = False
        cmbEWOffset.Enabled = False
        cmbNSOffset.Enabled = False
        txtUnitFactor.Enabled = False
       'DISable quartering related controls
        fraSubSection.Enabled = False
        cmbABCD.Enabled = False
        cmbNWofNE.Enabled = False
        optAsABCD.Enabled = False
        optAsNWofNE.Enabled = False







End Sub

Private Sub cmdCopyCoords_Click()
    CopyXYToClipboard
End Sub

Private Sub cmdFindCoordinates_Click()
    If strCurrentSection <> Trim$(UCase$(txtSectionName.Text)) Then Call cmdFindSection_Click
    If flgIHaveCurrentSection = False Then Exit Sub

    Dim flgItWorked As Long
    Select Case flgOffsetsFrom
      Case OFFSET_FROM_FOOTAGE
        'We have footages
            Dim dX As Single, dY As Single
         dX = Val(txtEWOffset): dY = Val(txtNSOffset)
         dX = dX * UnitMultiplier: dY = dY * UnitMultiplier
            ';picSection.Cls
            ';picSection.Circle (dX / 5280, dY), .5, RGB(255, 128, 0)
         flgItWorked = XYFrom_OutlineCornrOffst(sngSectionOutline(), CornerFrom, dX, dY, XOut,
YOut)

       Case OFFSET_FROM_NWofNE
         'We have subsections.  Use the NWNE representation to get coords
         Dim SubSect As String
```

```
        SubSect = txtSubNWNE
          'Make sure we have an even number of characters, and replace blanks with underscores
.
          If SubSect <> "" Then
            If Len(SubSect) Mod 2 Then SubSect = SubSect & " "
                Do While InStr(SubSect, " ")
                   Mid$(SubSect, InStr(SubSect, " "), 1) = "_"
                Loop
          End If
            Dim Col As Variant, Row As Variant
        Col = dXFromSubsectSEofNW(SubSect)
        Row = dYFromSubsectSEofNW(SubSect)
        If IsNull(Col) Or IsNull(Row) Then
             Dim MsgOut
             MsgOut = "Program could not determine how to handle the subdivision description "
 & SubSect & " ." & Chr$(13) & Chr$(10)
             MsgOut = MsgOut & "Please limit input to NW NE SW SE "
             Beep
             MsgBox MsgOut
             Exit Sub
        End If
        ColOffset = Col: RowOffset = Row
        flgItWorked = XYFromDXDYQuad(ColOffset, RowOffset, sngSectionOutline(), XOut, YOut)

      Case OFFSET_FROM_MAPOID
        'RowOffset and ColOffset were set by the picture_click or _dblClick events
        flgItWorked = XYFromDXDYQuad(ColOffset, RowOffset, sngSectionOutline(), XOut, YOut)
    End Select

   SetTextBoxColors RGB(255, 255, 255)

   txtX = XOut: txtY = YOut

End Sub

Private Sub cmdFindSection_Click()
   mousepointer = 11
   txtSubNWNE = "": txtSubsectionABD = ""
   txtEWOffset = "": txtNSOffset = ""
   txtX = "": txtY = ""

   strCurrentSection = Trim$(UCase$(txtSectionName.Text))

   flgIHaveCurrentSection = GetSectionCoords(strCurrentSection)
    flgIgnoreChanges = True

     txtSectionName = strCurrentSection
     txtSectionName.BackColor = RGB(255, 255, 255)
    flgIgnoreChanges = False

 ' Call GetCornersOfSection(sngSectionOutline(), NECorner, NWCorner, SWCorner, SECorner, BadCen
ter, DummyLong)

 ' Debug.Print SectionName
 ' DebugPrintRealPoint NECorner, "NECorner"
 ' DebugPrintRealPoint NWCorner, "NWCorner"
 ' DebugPrintRealPoint SWCorner, "SWCorner"
 ' DebugPrintRealPoint SECorner, "SECorner"
 ' DebugPrintRealPoint BadCenter, "BadCenter"



   mousepointer = 0
End Sub

Private Sub cmdFromDB_Click(Value As Integer)
    If DBName = "" Then
        Call GripeNoDatabase
        Exit Sub
```

```
    End If

    flgDataEntryStyle = DATA_ENTRY_BATCH
    Value = True
    pnlQTSubdivision.ZOrder
    cmdFromDB.ZOrder
    'zzMnuEdit.Enabled = False
End Sub

Private Sub cmdMANUAL_Click(Value As Integer)
    flgDataEntryStyle = DATA_ENTRY_MANUAL
    Value = True
    pnlHandEnteredLocations.ZOrder
    cmdManual.ZOrder
    'zzMnuEdit.Enabled = True
End Sub

Private Sub cmdStartConversion_Click()
'' converts legal descriptions for point locations to X,Y coordinates.

'First check for valid input state.
Dim strQuartersFrom As String, strFootageEWFrom As String, strCornersFrom As String, strFootage
NSFrom As String
Dim strSectionsFrom As String, ConversionFactor As Single, flgHowToGetLocation  As Long
Dim name_SectionsFrom As String, name_XTo As String, name_YTo As String
  Dim flgItWorked As Long


    If CheckQTInputs() = False Then Exit Sub
    'Add fields if need be.
    Call AddFieldsForOutput

'Get the names of all the fields, or possibly constants
    strSectionsFrom = cmbSectionName
    If optSubSection Then
        If optAsABCD Then
          flgHowToGetLocation = OFFSET_FROM_ABCD
          strQuartersFrom = cmbABCD
         Else
          flgHowToGetLocation = OFFSET_FROM_NWofNE
          strQuartersFrom = cmbNWofNE
        End If
        strQuartersFrom = StripQuotes(strQuartersFrom)
      Else 'At this point, either optSubsection or optFootage is true,
           ' so we can just grab the footage fields now.
        strFootageEWFrom = cmbEWOffset: strFootageEWFrom = StripQuotes(strFootageEWFrom)
        strFootageNSFrom = cmbNSOffset: strFootageNSFrom = StripQuotes(strFootageNSFrom)
        strCornersFrom = cmbFromCorner: strCornersFrom = StripQuotes(strCornersFrom)
        ConversionFactor = txtUnitFactor: If ConversionFactor = 0 Then ConversionFactor = 1
        flgHowToGetLocation = OFFSET_FROM_FOOTAGE
    End If
    name_XTo = Trim$(txtXFieldOut.Text)
    name_YTo = Trim$(txtYFieldOut.Text)

' and sweep through the dataset.

'Any constant should have a space in the leftmost location at this point.



Dim strCurSection As String, strCurSubSection As String
Dim strCurCorner As String, sngCurEW As Single, sngCurNS As Single
Dim XOut As Single, YOut As Single
   'Initialize these field values to the values of the "field name"
    ' If the field name starts with " ", then the big loop below
    ' will leave them alone, and we just use constants
    ' If the field name does contain a field name, we can retrieve it.
     strCurCorner = strCornersFrom
     sngCurEW = ConversionFactor * Val(strFootageEWFrom)
```

```
frmLegalToXY_DB_BOTH - 7

    sngCurNS = Val(strFootageEWFrom) * ConversionFactor
    strCurSubSection = strQuartersFrom

Dim nR As Long, nRecs As Long ', Temp As String
dat_SectionConversion.Recordset.MoveFirst: dat_SectionConversion.Recordset.MoveLast: dat_Sectio
nConversion.Recordset.MoveFirst
nRecs = dat_SectionConversion.Recordset.RecordCount

'Cheap error handling.  All problems resolved by not updating the record
On Error Resume Next
Err = 0
For nR = 1 To nRecs

    picPercentDone.Width = nR / nRecs * 100
    DoEvents

    strCurSection = dat_SectionConversion.Recordset(strSectionsFrom)
'Get the section outline, load coordinates into form level array sngXYData
    flgIHaveCurrentSection = False
    flgIHaveCurrentSection = GetSectionCoords(strCurSection)
    If flgIHaveCurrentSection = False Then GoTo SkipThisRecord
    Select Case flgHowToGetLocation
        Case OFFSET_FROM_FOOTAGE
            If Left$(strCornersFrom, 1) <> " " Then
                strCurCorner = dat_SectionConversion.Recordset(strCornersFrom)
            End If
            If Left$(strFootageEWFrom, 1) <> " " Then
                sngCurEW = ConversionFactor * dat_SectionConversion.Recordset(strFootageEWFrom)
            End If
            If Left$(strFootageNSFrom, 1) <> " " Then
                sngCurNS = ConversionFactor * dat_SectionConversion.Recordset(strFootageNSFrom)
            End If

            If sngCurEW = 0 And sngCurNS = 0 Then GoTo SkipThisRecord
            If Err <> 0 Then GoTo SkipThisRecord
            flgItWorked = XYFrom_OutlineCornrOffst(sngSectionOutline(), strCurCorner, sngCurEW,
 sngCurNS, XOut, YOut)

        Case OFFSET_FROM_NWofNE
            If Left$(strQuartersFrom, 1) <> " " Then
                strCurSubSection = FixSubSectionName(dat_SectionConversion.Recordset(strQuarter
sFrom))
            End If
            ColOffset = dXFromSubsectSEofNW(strCurSubSection)
            RowOffset = dYFromSubsectSEofNW(strCurSubSection)
            If Err <> 0 Then GoTo SkipThisRecord
            flgItWorked = XYFromDXDYQuad(ColOffset, RowOffset, sngSectionOutline(), XOut, YOut)

        Case OFFSET_FROM_ABCD
            If Left$(strQuartersFrom, 1) <> " " Then
                                'Do not condition ACDB for blanks
                strCurSubSection = dat_SectionConversion.Recordset(strQuartersFrom)
            End If
            ColOffset = dXFromABCD(strCurSubSection)
            RowOffset = dYFromSubSectABCD(strCurSubSection)
            If Err <> 0 Then GoTo SkipThisRecord
            flgItWorked = XYFromDXDYQuad(ColOffset, RowOffset, sngSectionOutline(), XOut, YOut)


    End Select
  If flgItWorked Then
    dat_SectionConversion.Recordset.Edit
    dat_SectionConversion.Recordset(name_XTo) = XOut
    dat_SectionConversion.Recordset(name_YTo) = YOut
    dat_SectionConversion.Recordset.Update
  End If
SkipThisRecord:
    Err = 0
    dat_SectionConversion.Recordset.MoveNext
```

```
Next nR

Beep
Beep
Beep
MsgBox "Done computing coordinates from legal descriptions."
End Sub

Private Sub CopyXYToClipboard()
    Clipboard.Clear
    Dim strXY As String
    strXY = txtX & Chr$(9) & txtY
    Clipboard.SetText strXY
End Sub

Private Function DescriptionOfInteractiveEntry() As String
    Dim strDescription As String
   Select Case flgOffsetsFrom
      Case OFFSET_FROM_FOOTAGE
         strDescription = txtEWOffset & " E/W, " & txtNSOffset & " N/S "
         strDescription = strDescription & " from " & CornerFrom & " Corner of " & strCurrentSe
ction
      Case OFFSET_FROM_NWofNE
         strDescription = "Center of " & txtSubNWNE & " of " & strCurrentSection
      Case OFFSET_FROM_MAPOID
         strDescription = "Mouse selected location "
   End Select
   strDescription = strDescription & " is located at (" & txtX & " , " & txtY & ")."
   DescriptionOfInteractiveEntry = strDescription
End Function

Private Sub DrawOffset()
    Dim dX As Single, dY As Single
    Const DARKRED = 128&


     dX = Val(txtEWOffset): dY = Val(txtNSOffset)
     dX = dX / 5280 * SECTION_STEP
     dY = dY / 5280 * SECTION_STEP

    picSection.Cls
    picSection.DrawWidth = 5



    Select Case CornerFrom
        Case "NW"
          dX = dX: dY = SECTION_STEP - dY
          picSection.Line (0, SECTION_STEP)-(dX, SECTION_STEP), DARKRED
          picSection.Line -(dX, dY), DARKRED
        Case "NE"
          dX = SECTION_STEP - dX: dY = SECTION_STEP - dY
          picSection.Line (SECTION_STEP, SECTION_STEP)-(dX, SECTION_STEP), DARKRED
          picSection.Line -(dX, dY), DARKRED

        Case "SW"
          dX = dX: dY = dY
          picSection.Line (0, 0)-(dX, 0), DARKRED
          picSection.Line -(dX, dY), DARKRED
        Case "SE"
          dX = SECTION_STEP - dX: dY = dY
          picSection.Line (SECTION_STEP, 0)-(dX, 0), DARKRED
          picSection.Line -(dX, dY), DARKRED

    End Select

    picSection.DrawWidth = 1
```

```
  picSection.Circle (dX, dY), 0.5, RGB(255, 0, 0)
End Sub

Private Sub Form_Load()
        CR = Chr$(13)
        TabChar = Chr$(9)
        CRLF = Chr$(13) & Chr$(10)
        ReDim DataPts(1 To 1)
        flgDataEntryStyle = DATA_ENTRY_MANUAL


On Error GoTo ERRForm_Load

 picSection.Scale (0, SECTION_STEP)-(SECTION_STEP, 0)
 UnitMultiplier = 0.3048
 zlblOffset = "No Corner Selected": CornerFrom = "   "
'Find out where the data is at from an INI File
'LegXY001.ini is laid out as a standard INI File.
'It contains the following information
'  Database name
'  name of query or table containing the section outlines
'     as WHEAT XYBLOBs
'  name of the field containing the KSSAS Serial numbers
'  name of the field containing the XYBLOBs

Dim strINIFile As String, strSectionTitle As String, strKeyName As String
Dim strBuffer As String, strDefaultValue As String, nBytesBack As Long

    strINIFile = app.Path & "\LegXY002.ini"
    strSectionTitle = "Legal To XY Conversion"
    strKeyName = "DatabaseWithUSPLSS_Sections"
    strDefaultValue = app.Path & "\usplss002.mdb"
    strBuffer = Space$(1024)
    nBytesBack = GetPrivateProfileString(strSectionTitle, ByVal strKeyName, strDefaultValue, st
rBuffer, 1024, strINIFile)
    SectionDBName = Left$(strBuffer, nBytesBack)

    Set SectionsDB = OpenDatabase(SectionDBName, False, False, "") 'Open the desired database


    strSectionTitle = "Fast Method"
    strKeyName = "ParameterQueryThatGetsOutlineBySerialNumber"
    strDefaultValue = ""
    strBuffer = Space$(1024)
    nBytesBack = GetPrivateProfileString(strSectionTitle, ByVal strKeyName, strDefaultValue, st
rBuffer, 1024, strINIFile)
    strParamQueryName = Left$(strBuffer, nBytesBack)


    strKeyName = "FieldContainingCoordinatesAsXYBLOB"
    strDefaultValue = ""
    strBuffer = Space$(1024)
    nBytesBack = GetPrivateProfileString(strSectionTitle, ByVal strKeyName, strDefaultValue, st
rBuffer, 1024, strINIFile)
    strBLOBField = Left$(strBuffer, nBytesBack)

    strKeyName = "DefaultSectionName"
    strDefaultValue = "13S19E02"    'Legal location of KGS
    strBuffer = Space$(1024)
    nBytesBack = GetPrivateProfileString(strSectionTitle, ByVal strKeyName, strDefaultValue, st
rBuffer, 1024, strINIFile)
    strCurrentSection = Left$(strBuffer, nBytesBack)


    If strBLOBField <> "" And strParamQueryName <> "" Then
            flgUseParameterQuery = True
            Set ParamQueryDef = SectionsDB.OpenQueryDef(strParamQueryName)
```

```
frmLegalToXY_DB_BOTH - 10

            If GetSectionCoords(strCurrentSection) Then
                flgUseParameterQuery = True
                txtSectionName = strCurrentSection
                Exit Sub
              Else
                flgUseParameterQuery = False
             End If
       End If


   flgUseParameterQuery = False

    strSectionTitle = "Slow Method"
    strKeyName = "TableOrQueryWithSectionOutlines"
    strDefaultValue = "KS Sections USPLSS"
    strBuffer = Space$(1024)
    nBytesBack = GetPrivateProfileString(strSectionTitle, ByVal strKeyName, strDefaultValue, st
rBuffer, 1024, strINIFile)
    strSectTable = Left$(strBuffer, nBytesBack)
    'Set the larger snapshot to the table (or query) containing the Sections
    Set snpLarge = SectionsDB.CreateSnapshot(strSectTable)
    snpLarge.MoveFirst

    strKeyName = "FieldContainingKSSASSerialNumber"
    strDefaultValue = "kssas_SerialNumber"
    strBuffer = Space$(1024)
    nBytesBack = GetPrivateProfileString(strSectionTitle, ByVal strKeyName, strDefaultValue, st
rBuffer, 1024, strINIFile)
    strSerialNumberField = Left$(strBuffer, nBytesBack)

    txtSectionName = kssas_TTdRRdSSFromSerialNumber(snpLarge(strSerialNumberField))

    strKeyName = "FieldContainingCoordinatesAsXYBLOB"
    strDefaultValue = "XYBLOB"
    strBuffer = Space$(1024)
    nBytesBack = GetPrivateProfileString(strSectionTitle, ByVal strKeyName, strDefaultValue, st
rBuffer, 1024, strINIFile)
    strBLOBField = Left$(strBuffer, nBytesBack)

    strCurrentSection = Trim$(UCase$(txtSectionName))
    DummyLong = GetSectionCoords(strCurrentSection)



Exit Sub
ERRForm_Load:
 Select Case Err
  Case Else
    MsgBox "Error # " & Err & " occurred trying to start LegalCoordinates" & Chr$(13) & Chr(10)
 & Error
    Err = 0
    Exit Sub
    Resume
 End Select



End Sub

Private Function GetSectionCoords(SectionName As String) As Long
If flgIHaveCurrentSection Then Exit Function
flgOffsetsFrom = OFFSET_FROM_NWofNE

On Error GoTo ERRGetSectionCoords

 lngSectionSerialNumber = kssas_SerialNumberFromTTdRRdSS(SectionName)
 'The change for version 3 is that instead of the section serial number
 ' ,which is somewhat processor dependent, we'll use the row and column number
 intColForSection = kssas_ColumnFromTTdRRdSS(SectionName)
```

```
 intRowForSection = kssas_RowFromTTdRRdSS(SectionName)



 If flgUseParameterQuery Then
    ParamQueryDef!SerialNumberWanted = lngSectionSerialNumber ' Set parameter.
   ' ParamQueryDef!ColumnWanted = intColForSection ' Set parameter.
   ' ParamQueryDef!RowWanted = intRowForSection ' Set parameter.
    Set snpSmall = ParamQueryDef.CreateSnapshot()              ' Create Snapshot.
   Else 'use slow method
    snpLarge.Filter = strSerialNumberField & " = " & lngSectionSerialNumber
    Set snpSmall = snpLarge.CreateSnapshot()
 End If
 snpSmall.MoveFirst '

 strTemp = snpSmall(strBLOBField)

 Erase sngSectionOutline
 Call BLOB2XYPairsInSnglArray(strTemp, sngSectionOutline())
 flgIHaveCurrentSection = True: GetSectionCoords = True
 snpSmall.Close
 Set snpSmall = Nothing

Exit Function



ERRGetSectionCoords:
Select Case Err
 Case 3021 'No current records
      Beep
      If flgDataEntryStyle = DATA_ENTRY_MANUAL Then
         MsgBox "Section " & SectionName & " was not found.  It appears it is not in the curren
t database."
      End If
      GetSectionCoords = False: flgIHaveCurrentSection = False
      Err = 0
      Exit Function
      Resume
 Case 94
   'Invalid use of null, probably a null returned by SerialNumberFromTTdRRdSS
      Beep
      If flgDataEntryStyle = DATA_ENTRY_MANUAL Then
         MsgBox "Section " & SectionName & " is not valid section name.  Please re-enter."
      End If
      GetSectionCoords = False: flgIHaveCurrentSection = False
      Err = 0
      Exit Function

 Case Else
 'GWPouch  Made the messagebox conditional 96 04 02.
 ' It was very disturbing to run into this in a loop.
    Beep
    If flgDataEntryStyle = DATA_ENTRY_MANUAL Then
      MsgBox "Error # " & Err & " occurred trying to Get Section Coordinates." & Chr$(13) & Chr
(10) & Error

    End If
    Err = 0
    GetSectionCoords = False: flgIHaveCurrentSection = False
    Exit Function
    Resume
 End Select

End Function

Private Sub GripeNoDatabase()
    Call cmdMANUAL_Click(0)
    cmdManual = True
```

```
    Beep
    MsgBox "You cannot use this program to convert USPLSS legal descriptions stored in a databa
se to coordinates until you have opened a database."

   Call mfOpenDB_Click
End Sub

Private Sub ListQrysTblsInComboBox(DBIn As Database, ComboOut As ComboBox)
' this subroutine lists all queries and tables in a database
'   in a combobox, for later user selection.
    ComboOut.Clear
    Dim DB_SYSTEMOBJECT As Long
    DB_SYSTEMOBJECT = &H80000002

    Dim snpQTList As Snapshot 'declared locally so it goes away
    Set snpQTList = DBIn.ListTables()

    Dim QTName As String
    snpQTList.MoveFirst
    Do Until snpQTList.EOF 'For each Table and query
        If Not ((snpQTList("Attributes") And DB_SYSTEMOBJECT) <> 0) And (snpQTList("Attributes"
) < 6) Then
          'Exclude system objects from list of tables and queries presented to user
          QTName = snpQTList("Name")
          ComboOut.AddItem QTName
        End If
        snpQTList.MoveNext
    Loop
    snpQTList.Close
    Set snpQTList = Nothing
End Sub

Private Sub meCopy_Click()
Dim strClipOut As String
   Clipboard.Clear
      If ActiveControl Is cmbAllFields Then strClipOut = cmbAllFields
      If TypeOf ActiveControl Is TextBox Then strClipOut = ActiveControl.Text
      If ActiveControl Is txtX Then strClipOut = txtX.Text & TabChar & txtY.Text & CRLF
   Clipboard.SetText strClipOut
End Sub

Private Sub mePaste_Click()
Dim strClipIn As String, str1 As String, str2 As String, T1 As Long, T2 As Long
   strClipIn = Clipboard.GetText()

   If TypeOf ActiveControl Is TextBox Then
       T1 = InStr(strClipIn, TabChar)
       If T1 = 0 Then
            ActiveControl = strClipIn
            Exit Sub
       End If
       str1 = Left$(strClipIn, T1 - 1)
       T2 = InStr(strClipIn, CRLF)
       If T2 = 0 Then T2 = InStr(strClipIn, Chr$(10))
       If T2 = 0 Then T2 = InStr(strClipIn, Chr$(13))
       If T2 = 0 Then T2 = Len(strClipIn) + 1
       str2 = Mid$(strClipIn, T1 + 1, T2 - T1 - 1)
    Else
      If TypeOf ActiveControl Is ComboBox Then
        ActiveControl.AddItem strClipIn
        ActiveControl.ListIndex = ActiveControl.ListCount - 1
      End If
   End If



End Sub
```

```
Private Sub mfCloseDatabase_Click()
 On Error Resume Next
 DB.Close
  DBName = ""
  Me.Caption = "LLUTMnn_  UTM<=>Longitude-Latitude Projection " & "NO DATABASE OPEN"
 Set DB = Nothing

  dat_SectionConversion.Recordset.Close
  cmbQTList_SubSection.Clear
  cmbSectionName.Clear


End Sub

Private Sub mfExit_Click()
 End
End Sub

Private Sub mfOpenDB_Click()
On Error GoTo ERRmfOpenDB
CMDialog1.DialogTitle = "Database to Open"
CMDialog1.Filter = "MS Access Databases|*.mdb"
CMDialog1.Flags = OFN_FILEMUSTEXIST
CMDialog1.Action = 1
CMDialog1.CancelError = True
Dim Temp As String

Temp = CMDialog1.Filename
If Len(Temp) = 0 Then 'the user did not enter a file name
    Else
     DBName = Temp
     '  Open an MSAccess database for non-exclusive, read-write access.
     Set DB = OpenDatabase(DBName, False, False, "") 'Open the desired database

     Me.Caption = "LegXY002  Legal Description to X,Y Conversion " & DBName
     dat_SectionConversion.DatabaseName = DBName
     Call ListQrysTblsInComboBox(DB, cmbQTList_SubSection)
        pnlQTSubdivision.ZOrder
        cmdFromDB.ZOrder
        cmdFromDB = True


         cmbQTList_SubSection.SetFocus
         SendKeys "%{DOWN}"
         DoEvents
End If
Exit Sub
ERRmfOpenDB:
 Select Case Err
   Case Else
   Exit Sub
   Resume
 End Select
End Sub

Private Sub optFootage_Click(Value As Integer)
    If Value Then
      'Enable footage related controls
        fraFootage.Enabled = True
        cmbFromCorner.Enabled = True
        cmbEWOffset.Enabled = True
        cmbNSOffset.Enabled = True
        txtUnitFactor.Enabled = True
       'Disable quartering related controls
        fraSubSection.Enabled = False
        cmbABCD.Enabled = False
        cmbNWofNE.Enabled = False
        optAsABCD.Enabled = False
        optAsNWofNE.Enabled = False
```

```
    End If

End Sub

Private Sub optNE_Click(Value As Integer)
 CornerFrom = "NE"
 zlblOffset = "Offsets from NE corner."
 ClearSubSections
 flgOffsetsFrom = OFFSET_FROM_FOOTAGE
End Sub

Private Sub optNW_Click(Value As Integer)
     CornerFrom = "NW"
     zlblOffset = "Offsets from NW corner."
     ClearSubSections
     flgOffsetsFrom = OFFSET_FROM_FOOTAGE
End Sub

Private Sub optSE_Click(Value As Integer)
    CornerFrom = "SE"
    zlblOffset = "Offsets from SE corner."
    ClearSubSections
    flgOffsetsFrom = OFFSET_FROM_FOOTAGE
End Sub

Private Sub optSubSection_Click(Value As Integer)
    If Value Then
      'DISable footage related controls
        fraFootage.Enabled = False
        cmbFromCorner.Enabled = False
        cmbEWOffset.Enabled = False
        cmbNSOffset.Enabled = False
        txtUnitFactor.Enabled = False
       'ENsable quartering related controls
        fraSubSection.Enabled = True
        cmbABCD.Enabled = True
        cmbNWofNE.Enabled = True
        optAsABCD.Enabled = True
        optAsNWofNE.Enabled = True
    End If


End Sub

Private Sub optSW_Click(Value As Integer)
   CornerFrom = "SW"
   zlblOffset = "Offsets from SW corner."
   ClearSubSections
   flgOffsetsFrom = OFFSET_FROM_FOOTAGE
End Sub

Private Sub ParseABDSubSect(ABCD As String, FractionEast As Single, FractionNorth As Single)
Dim NWNE As String
ABCD = UCase$(ABCD)
Dim nQ As Integer, nQrtrs As Integer
nQrtrs = Len(ABCD)
For nQ = 1 To nQrtrs
    Select Case Mid$(ABCD, nQ, 1)
        Case "A"
            NWNE = "NE" & NWNE
        Case "B"
            NWNE = "NW" & NWNE
        Case "C"
            NWNE = "SW" & NWNE
        Case "D"
            NWNE = "SE" & NWNE
        Case Else
            Beep
            Exit Sub
```

```
      End Select
Next nQ
txtSubNWNE = NWNE
ParseNESESubSect NWNE, FractionEast, FractionNorth
End Sub

Private Sub ParseNESESubSect(ByVal Quarters As String, FractionEast As Single, FractionNorth As
 Single)
'940926 GWPouch KGS.
'This subroutine takes a text string in the form NWNESE, where SE is the largest quartering and
 NW is the smallest quartering
' and finds the fraction of the distance across the section, from the SW corner.
'      |------------------------ |
'      |                         |
'      |                         |
'      |                         |
'      |                         |
'      |              ___________|
'      |              |       | X |
'      |              |       |   |
'      |              |       ------|
'      |______________|_____________|


'Initialize to center of section
Dim LenString As Integer, nQ As Integer, nQrtrs As Integer
LenString = Len(Quarters): Quarters = UCase$(Quarters)
nQrtrs = LenString / 2

FractionEast = 0.5
For nQ = 1 To nQrtrs
    If Mid$(Quarters, (nQrtrs - nQ + 1) * 2, 1) = "E" Then
        FractionEast = FractionEast + 2 ^ (-nQ - 1)
    End If
    If Mid$(Quarters, (nQrtrs - nQ + 1) * 2, 1) = "W" Then
        FractionEast = FractionEast - 2 ^ (-nQ - 1)
    End If
Next nQ


FractionNorth = 0.5
For nQ = 1 To nQrtrs
    If Mid$(Quarters, (nQrtrs - nQ + 1) * 2 - 1, 1) = "N" Then
        FractionNorth = FractionNorth + 2 ^ (-nQ - 1)
    End If
    If Mid$(Quarters, (nQrtrs - nQ + 1) * 2 - 1, 1) = "S" Then
        FractionNorth = FractionNorth - 2 ^ (-nQ - 1)
    End If
Next nQ


    ColOffset = FractionEast * SECTION_STEP
    RowOffset = FractionNorth * SECTION_STEP
    picSection.Cls
    picSection.Circle (ColOffset, RowOffset), 0.5, RGB(255, 128, 0)
    flgOffsetsFrom = OFFSET_FROM_NWofNE

End Sub

Private Sub picSection_Click()

     ColOffset = MouseX: RowOffset = MouseY
     picSection.Cls
     picSection.Circle (ColOffset, RowOffset), 0.5, RGB(255, 128, 0)
     flgOffsetsFrom = OFFSET_FROM_MAPOID
     'txtX = ColOffset: txtY = RowOffset
     ClearFootages
     ClearSubSections
End Sub
```

```
Private Sub picSection_DblClick()
     ColOffset = Int(MouseX / 4 + 0.5) * 4: RowOffset = Int(MouseY / 4 + 0.5) * 4

     picSection.Cls
     picSection.Circle (ColOffset, RowOffset), 0.5, RGB(255, 255, 0)
     flgOffsetsFrom = OFFSET_FROM_MAPOID
     'txtX = ColOffset: txtY = RowOffset
     ClearFootages
     ClearSubSections
End Sub

Private Sub picSection_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
     MouseX = X: MouseY = Y
End Sub

Private Sub pnlHandEnteredLocations_DblClick()
    Clipboard.Clear
    Clipboard.SetText DescriptionOfInteractiveEntry()
End Sub

Private Sub SetQuery_Points(strTblQrySQL As String)
'This subroutine sets the data control to the desired TaBLe, QueRY, or SQL string,
' then sets the labels for the attribute grid.
On Error GoTo ERRSetQuery
Set dnsCurrentQT = DB.CreateDynaset(strTblQrySQL)
nFields = dnsCurrentQT.Fields.Count
dnsCurrentQT.Close
Set dnsCurrentQT = Nothing
dat_SectionConversion.RecordSource = strTblQrySQL
        'nFields = dat_SectionConversion.Recordset.Fields.Count
dat_SectionConversion.Refresh
intCoordinateField = 0
Dim FieldName As String, uFieldName As String
nFields = dat_SectionConversion.Recordset.Fields.Count
Dim intHasX As Integer, intHasY As Integer, intHasUTM As Integer, intHasLL As Integer
'cmbXY_LongLat_Points(X_INDEX).Clear : cmbXY_LongLat_Points(Y_INDEX).Clear
'cmbXY_UTM_Points(X_INDEX).Clear : cmbXY_UTM_Points(Y_INDEX).Clear

For n = 0 To nFields - 1
    If ((dat_SectionConversion.Recordset.Fields(n).Type = DB_SINGLE) Or (dat_SectionConversion.
Recordset.Fields(n).Type = DB_DOUBLE)) Then    ' field might contain coordinates
      intHasX = False: intHasY = False: intHasUTM = False: intHasLL = False
      FieldName = dat_SectionConversion.Recordset.Fields(n).Name: uFieldName = UCase$(FieldName
)

      intHasX = intHasX Or InStr(uFieldName, "X_") <> 0
      intHasX = intHasX Or InStr(uFieldName, "EAST") <> 0

      intHasY = intHasY Or InStr(uFieldName, "Y_") <> 0
      intHasY = intHasY Or InStr(uFieldName, "NORT") <> 0


      intHasUTM = InStr(uFieldName, "UTM") <> 0

      intHasLL = intHasLL Or InStr(uFieldName, "ITUDE") <> 0
      intHasLL = intHasLL Or InStr(uFieldName, "LL") <> 0
      intHasLL = intHasLL Or InStr(uFieldName, "GEO") <> 0
      intHasLL = intHasLL Or InStr(uFieldName, "LON") <> 0
      intHasLL = intHasLL Or InStr(uFieldName, "LAT") <> 0

 '     If (intHasX And intHasUTM) Then cmbXY_UTM_Points(X_INDEX).AddItem FieldName
 '     If (intHasY And intHasUTM) Then cmbXY_UTM_Points(Y_INDEX).AddItem FieldName
 '
 '     If (intHasX And intHasLL) Then cmbXY_LongLat_Points(X_INDEX).AddItem FieldName
 '     If (intHasY And intHasLL) Then cmbXY_LongLat_Points(Y_INDEX).AddItem FieldName

    End If

Next n
```

```
'If cmbXY_UTM_Points(X_INDEX).ListCount = 0 And cmbXY_LongLat_Points(X_INDEX).ListCount = 0 The
n
'  Beep
'  MsgBox "That query or table does not contain any suitable coordinate fields.  Please choose
another table/query."
'  Exit Sub
'End If

'If cmbXY_LongLat_Points(X_INDEX).ListCount > 0 Then
'    cmbXY_LongLat_Points(X_INDEX).ListIndex = 0
'   Else
' '   cmbXY_LongLat_Points(X_INDEX).Text = "X_Longitude"
'End If
'If cmbXY_LongLat_Points(Y_INDEX).ListCount > 0 Then
'    cmbXY_LongLat_Points(Y_INDEX).ListIndex = 0
'   Else
'    cmbXY_LongLat_Points(Y_INDEX).Text = "Y_Latitude"
'End If
'

'If cmbXY_UTM_Points(X_INDEX).ListCount > 0 Then
'    cmbXY_UTM_Points(X_INDEX).ListIndex = 0
'   Else
'    cmbXY_UTM_Points(X_INDEX).Text = "X_Easting_UTM" & Format$(jUTMZone, "00")
'End If
'If cmbXY_UTM_Points(Y_INDEX).ListCount > 0 Then
'    cmbXY_UTM_Points(Y_INDEX).ListIndex = 0
'   Else
'    cmbXY_UTM_Points(Y_INDEX).Text = "Y_Northing_UTM" & Format$(jUTMZone, "00")
'End If


Exit Sub

ERRSetQuery:
If Err = 3061 Then
    MsgBox "You tried selecting a parameter query, and this application doesn't do those.  Sorr
y!"
    Err = 0
    Exit Sub
End If

MsgBox "Error number " & Err & " occurred in SetQuery_Sections." & CRLF & Error
Err = 0
Exit Sub
Resume
'dat_SectionConversion.Recordset.MoveFirst
'Eliminated the above line so empty tables wouldn't
'  cause it to freak out.
End Sub

Private Sub SetQuery_Sections(strTblQrySQL As String)
''This suboutine gets a list of all fields int the current table or query
'' and adds them to the list boxes, based on type and substring recognition
'Greg Pouch 95 03 18  AT HOME  About 5 hours.

On Error GoTo ERRSetQuery_Sections
    Set dnsCurrentQT = DB.CreateDynaset(strTblQrySQL)
    nFields = dnsCurrentQT.Fields.Count
    dnsCurrentQT.Close
    Set dnsCurrentQT = Nothing
    dat_SectionConversion.RecordSource = strTblQrySQL

    dat_SectionConversion.Refresh


    nFields = dat_SectionConversion.Recordset.Fields.Count
```

```
    cmbSectionName.Clear: cmbAllFields.Clear
    cmbABCD.Clear: cmbNWofNE.Clear
    cmbEWOffset.Clear: cmbNSOffset.Clear: cmbFromCorner.Clear

Dim FieldName As String, uFieldName As String, intFieldType As Integer
Dim flgSection As Integer, flgSubSection As Integer, flgEasting As Integer
Dim flgNorthing As Integer, flgCorner As Integer

For n = 0 To nFields - 1
    flgSection = False: flgSubSection = False: flgEasting = False: flgNorthing = False: flgCorn
er = False
    FieldName = dat_SectionConversion.Recordset.Fields(n).Name: uFieldName = UCase$(FieldName)
    cmbAllFields.AddItem FieldName

    intFieldType = dat_SectionConversion.Recordset.Fields(n).Type

    flgSection = InStr(uFieldName, "SECTION") Or InStr(uFieldName, "TTDRRDSS") Or InStr(uFieldN
ame, "USPLSS")
    flgSubSection = InStr(uFieldName, "SUB") Or InStr(uFieldName, "QUARTER") Or InStr(uFieldNam
e, "DIV")
    flgEasting = InStr(uFieldName, "EAST") Or InStr(uFieldName, "X_") Or InStr(uFieldName, "WES
T") Or InStr(uFieldName, "OFFSET")
    flgNorthing = InStr(uFieldName, "NORTH") Or InStr(uFieldName, "SOUTH") Or InStr(uFieldName,
 "Y_")
    flgCorner = InStr(uFieldName, "CORNER") Or InStr(uFieldName, "FROM")



    If (intFieldType = DB_TEXT) And flgSection Then cmbSectionName.AddItem FieldName
    If (intFieldType = DB_TEXT) And flgSubSection Then cmbABCD.AddItem FieldName: cmbNWofNE.Add
Item FieldName

    If (intFieldType = DB_TEXT) And flgCorner Then cmbFromCorner.AddItem FieldName
    If ((intFieldType = DB_SINGLE) Or (intFieldType = DB_DOUBLE)) And flgNorthing Then cmbNSOff
set.AddItem FieldName
    If ((intFieldType = DB_SINGLE) Or (intFieldType = DB_DOUBLE)) And flgEasting Then cmbEWOffs
et.AddItem FieldName
Next n
cmbAllFields.ListIndex = 0
cmbFromCorner.AddItem """SW""": cmbFromCorner.AddItem """NW""": cmbFromCorner.AddItem """NE""":
 cmbFromCorner.AddItem """SE"""
cmbFromCorner.ListIndex = 0

cmbNSOffset.AddItem """0.00001""": cmbNSOffset.ListIndex = 0
cmbEWOffset.AddItem """0.00001""": cmbEWOffset.ListIndex = 0

If cmbABCD.ListCount > 0 Then cmbABCD.ListIndex = 0
If cmbNWofNE.ListCount > 0 Then cmbNWofNE.ListIndex = 0

'xxx Check for empty lists
If cmbSectionName.ListCount > 0 Then
    cmbSectionName.ListIndex = 0
   Else
    Beep
    MsgBox "Program was unable to find a field name that looked like it contains section names.
  Please find the field containing Section Names in TTdRRdSS format in the list of all fields,
copy the field name, and paste it into the list  Field containing section name as TTdRRdSS
 or Choose a new table or query."
    cmbAllFields.SetFocus
    SendKeys "%{DOWN}"
    Exit Sub
End If




Exit Sub
```

```
ERRSetQuery_Sections:
If Err = 3061 Then
    MsgBox "You tried selecting a parameter query, and this application doesn't do those.  Sorr
y!"
    Err = 0
    Exit Sub
End If

MsgBox "Error number " & Err & " occurred in SetQuery_Sections." & CRLF & Error
Err = 0
Exit Sub
Resume
'dat_SectionConversion.Recordset.MoveFirst
'Eliminated the above line so empty tables wouldn't
'  cause it to freak out.
End Sub

Private Sub SetTextBoxColors(Color As Long)
        txtEWOffset.BackColor = Color
        txtNSOffset.BackColor = Color
        txtSectionName.BackColor = Color
        txtSubNWNE.BackColor = Color
        txtSubsectionABD.BackColor = Color
End Sub

Private Function StripQuotes(ByVal InString As String) As String
 Static Temp As String, iLoc As Integer
 Temp = InString
 iLoc = InStr(Temp, DOUBLEQUOTE)
 If iLoc Then
    Temp = Left$(Temp, iLoc - 1) & " " & Mid$(Temp, iLoc + 1)
    Temp = StripQuotes(Temp)
 End If
 StripQuotes = Temp

End Function

Private Sub txtEWOffset_Change()
    If flgIgnoreChanges Then Exit Sub
    If CornerFrom = "   " Then
        MsgBox "You need to check a box at one of the corners so the program knows where the of
fsets are from."
        Exit Sub
      End If

    txtEWOffset.BackColor = QBColor(11)
    ClearSubSections
    flgOffsetsFrom = OFFSET_FROM_FOOTAGE
    DrawOffset
End Sub

Private Sub txtNSOffset_Change()
    If flgIgnoreChanges Then Exit Sub
    If CornerFrom = "   " Then
        MsgBox "You need to check a box at one of the corners so the program knows where the of
fsets are from."
        Exit Sub
    End If

    txtNSOffset.BackColor = QBColor(11)
    ClearSubSections
    flgOffsetsFrom = OFFSET_FROM_FOOTAGE
    DrawOffset
End Sub

Private Sub txtOffsetMultiplier_Change()
   UnitMultiplier = Val(txtOffsetMultiplier)
End Sub
```

```
Private Sub txtSectionName_Change()
  If flgIgnoreChanges Then Exit Sub
  flgIHaveCurrentSection = False
  txtSectionName.BackColor = QBColor(11)
End Sub

Private Sub txtSubNWNE_Change()
   If flgIgnoreChanges Then Exit Sub
   txtSubNWNE.BackColor = QBColor(11)
   ClearFootages
   flgOffsetsFrom = OFFSET_FROM_NWofNE

   Dim NWNE As String
   Dim FractEast As Single, FractNorth As Single
   NWNE = txtSubNWNE
   Call ParseNESESubSect(NWNE, FractEast, FractNorth)

End Sub

Private Sub txtSubsectionABD_Change()
   If flgIgnoreChanges Then Exit Sub
   txtSubsectionABD.BackColor = QBColor(11)

   ClearFootages
   flgOffsetsFrom = OFFSET_FROM_NWofNE

   Dim ABCD As String
   Dim FractEast As Single, FractNorth As Single
   ABCD = txtSubsectionABD
   Call ParseABDSubSect(ABCD, FractEast, FractNorth)


End Sub

Private Sub txtX_DblClick()
   Call CopyXYToClipboard
End Sub
```

```
VERSION 4.00
Begin VB.Form frmLegalToXY_DB_BOTH
   Appearance      =   0  'Flat
   BackColor       =   &H00C0C0C0&
   Caption         =   "Leg XY Legal Description to Approximate XY Conversion"
   ClientHeight    =   5580
   ClientLeft      =   195
   ClientTop       =   1020
   ClientWidth     =   8970
   BeginProperty Font
      name            =   "MS Sans Serif"
      charset         =   1
      weight          =   700
      size            =   8.25
      underline       =   0   'False
      italic          =   0   'False
      strikethrough   =   0   'False
   EndProperty
   ForeColor       =   &H80000008&
   Height          =   6270
   Icon            =   (Icon)
   Left            =   135
   LinkTopic       =   "Form1"
   ScaleHeight     =   372
   ScaleMode       =   3  'Pixel
   ScaleWidth      =   598
   Top             =   390
   Width           =   9090
   Begin Threed.SSPanel pnlHandEnteredLocations
      Height          =   4680
      Left            =   150
      TabIndex        =   23
      Top             =   750
      Width           =   8700
      _version        =   65536
      _extentx        =   15346
      _extenty        =   8255
      _stockprops     =   15
      caption         =   "  Hand Entered Locations"
      bevelwidth      =   2
      borderwidth     =   0
      outline         =   -1  'True
      alignment       =   0
      autosize        =   3
      Begin Threed.SSFrame zfrmSubsection
         Height          =   1035
         Left            =   4605
         TabIndex        =   43
         Top             =   1080
         Width           =   3600
         _version        =   65536
         _extentx        =   6350
         _extenty        =   1826
         _stockprops     =   14
         caption         =   "By Subsection"
         forecolor       =   0
         BeginProperty font {FB8F0823-0164-101B-84ED-08002B2EC713}
            name            =   "MS Sans Serif"
            charset         =   1
            weight          =   400
            size            =   8.25
            underline       =   0   'False
            italic          =   0   'False
            strikethrough   =   0   'False
         EndProperty
         Begin VB.TextBox txtSubsectionABD
            Appearance      =   0  'Flat
            BeginProperty Font
               name            =   "MS Sans Serif"
```

```
            charset         =   1
            weight          =   400
            size            =   8.25
            underline       =   0   'False
            italic          =   0   'False
            strikethrough   =   0   'False
         EndProperty
         Height          =   300
         Left            =   120
         TabIndex        =   45
         Top             =   660
         Width           =   1100
      End
      Begin VB.TextBox txtSubNWNE
         Appearance      =   0  'Flat
         BeginProperty Font
            name            =   "MS Sans Serif"
            charset         =   1
            weight          =   400
            size            =   8.25
            underline       =   0   'False
            italic          =   0   'False
            strikethrough   =   0   'False
         EndProperty
         Height          =   300
         Left            =   120
         TabIndex        =   44
         Top             =   300
         Width           =   1100
      End
      Begin VB.Label lblSubADCB
         Appearance      =   0  'Flat
         BackColor       =   &H80000005&
         BackStyle       =   0  'Transparent
         Caption         =   "in ABD format"
         BeginProperty Font
            name            =   "MS Sans Serif"
            charset         =   1
            weight          =   400
            size            =   8.25
            underline       =   0   'False
            italic          =   0   'False
            strikethrough   =   0   'False
         EndProperty
         ForeColor       =   &H80000008&
         Height          =   255
         Left            =   1260
         TabIndex        =   47
         Top             =   660
         Width           =   1095
      End
      Begin VB.Label zlblSubNEofNW
         Appearance      =   0  'Flat
         BackColor       =   &H80000005&
         BackStyle       =   0  'Transparent
         Caption         =   "in NW 1/4 of NE 1/4 format"
         BeginProperty Font
            name            =   "MS Sans Serif"
            charset         =   1
            weight          =   400
            size            =   8.25
            underline       =   0   'False
            italic          =   0   'False
            strikethrough   =   0   'False
         EndProperty
         ForeColor       =   &H80000008&
         Height          =   255
         Left            =   1260
         TabIndex        =   46
```

```
      Top             =   300
      Width           =   2055
   End
End
Begin VB.CommandButton cmdFindCoordinates
   Appearance      =   0  'Flat
   BackColor       =   &H80000005&
   Caption         =   "Find Coordinates"
   Default         =   -1  'True
   Height          =   300
   Left            =   6400
   TabIndex        =   42
   Top             =   4120
   Width           =   1800
End
Begin Threed.SSFrame ZFrameFootage
   Height          =   795
   Left            =   4605
   TabIndex        =   34
   Top             =   2220
   Width           =   3600
   _version        =   65536
   _extentx        =   6350
   _extenty        =   1402
   _stockprops     =   14
   caption         =   "By Offset"
   BeginProperty font {FB8F0823-0164-101B-84ED-08002B2EC713}
      name            =   "MS Sans Serif"
      charset         =   1
      weight          =   400
      size            =   8.25
      underline       =   0   'False
      italic          =   0   'False
      strikethrough   =   0   'False
   EndProperty
   Begin VB.TextBox txtOffsetMultiplier
      Appearance      =   0  'Flat
      BackColor       =   &H00C0C0C0&
      BorderStyle     =   0  'None
      BeginProperty Font
         name            =   "MS Sans Serif"
         charset         =   1
         weight          =   400
         size            =   8.25
         underline       =   0   'False
         italic          =   0   'False
         strikethrough   =   0   'False
      EndProperty
      Height          =   220
      Left            =   2760
      TabIndex        =   37
      Text            =   "0.3048"
      Top             =   480
      Width           =   735
   End
   Begin VB.TextBox txtNSOffset
      Appearance      =   0  'Flat
      BeginProperty Font
         name            =   "MS Sans Serif"
         charset         =   1
         weight          =   400
         size            =   8.25
         underline       =   0   'False
         italic          =   0   'False
         strikethrough   =   0   'False
      EndProperty
      Height          =   300
      Left            =   2820
      TabIndex        =   36
```

```
      Top             =   180
      Width           =   675
   End
   Begin VB.TextBox txtEWOffset
      Appearance      =   0  'Flat
      BeginProperty Font
         name            =   "MS Sans Serif"
         charset         =   1
         weight          =   400
         size            =   8.25
         underline       =   0   'False
         italic          =   0   'False
         strikethrough   =   0   'False
      EndProperty
      Height          =   300
      Left            =   1800
      TabIndex        =   35
      Top             =   180
      Width           =   615
   End
   Begin VB.Label zlblUnitMultiplier
      Appearance      =   0  'Flat
      BackColor       =   &H80000005&
      BackStyle       =   0  'Transparent
      Caption         =   "Unit Multiplier"
      BeginProperty Font
         name            =   "MS Sans Serif"
         charset         =   1
         weight          =   400
         size            =   8.25
         underline       =   0   'False
         italic          =   0   'False
         strikethrough   =   0   'False
      EndProperty
      ForeColor       =   &H80000008&
      Height          =   255
      Index           =   0
      Left            =   1560
      TabIndex        =   41
      Top             =   480
      Width           =   1275
   End
   Begin VB.Label ZlblNS
      Appearance      =   0  'Flat
      BackColor       =   &H80000005&
      BackStyle       =   0  'Transparent
      Caption         =   "NS"
      BeginProperty Font
         name            =   "MS Sans Serif"
         charset         =   1
         weight          =   400
         size            =   8.25
         underline       =   0   'False
         italic          =   0   'False
         strikethrough   =   0   'False
      EndProperty
      ForeColor       =   &H80000008&
      Height          =   255
      Left            =   2520
      TabIndex        =   40
      Top             =   180
      Width           =   315
   End
   Begin VB.Label ZlblEW
      Appearance      =   0  'Flat
      BackColor       =   &H80000005&
      BackStyle       =   0  'Transparent
      Caption         =   "EW"
      BeginProperty Font
```

```
            name            =   "MS Sans Serif"
            charset         =   1
            weight          =   400
            size            =   8.25
            underline       =   0   'False
            italic          =   0   'False
            strikethrough   =   0   'False
         EndProperty
         ForeColor       =   &H80000008&
         Height          =   195
         Left            =   1440
         TabIndex        =   39
         Top             =   180
         Width           =   375
      End
      Begin VB.Label zlblOffset
         Appearance      =   0  'Flat
         BackColor       =   &H80000005&
         BackStyle       =   0  'Transparent
         Caption         =   "Offsets from SW corner"
         BeginProperty Font
            name            =   "MS Sans Serif"
            charset         =   1
            weight          =   400
            size            =   8.25
            underline       =   0   'False
            italic          =   0   'False
            strikethrough   =   0   'False
         EndProperty
         ForeColor       =   &H80000008&
         Height          =   435
         Left            =   120
         TabIndex        =   38
         Top             =   300
         Width           =   1275
      End
   End
   Begin VB.TextBox txtX
      Appearance      =   0  'Flat
      Height          =   300
      Left            =   4740
      TabIndex        =   29
      Top             =   3300
      Width           =   1500
   End
   Begin VB.TextBox txtY
      Appearance      =   0  'Flat
      Height          =   300
      Left            =   6700
      TabIndex        =   28
      Top             =   3300
      Width           =   1500
   End
   Begin VB.CommandButton cmdFindSection
      Appearance      =   0  'Flat
      BackColor       =   &H80000005&
      Caption         =   "&Find Section"
      Height          =   300
      Left            =   6405
      TabIndex        =   27
      Top             =   660
      Width           =   1800
   End
   Begin VB.TextBox txtSectionName
      Appearance      =   0  'Flat
      Height          =   300
      Left            =   4740
      TabIndex        =   26
      Text            =   "10S21W03"
```

```
      Top             =   660
      Width           =   1455
   End
   Begin Threed.SSPanel zpnlHostPicture
      Height          =   3975
      Left            =   360
      TabIndex        =   24
      Top             =   450
      Width           =   3975
      _version        =   65536
      _extentx        =   7011
      _extenty        =   7011
      _stockprops     =   15
      caption         =   "pnlHostForMapoid"
      borderwidth     =   2
      bevelinner      =   1
      autosize        =   3
      Begin VB.PictureBox picSection
         Appearance      =   0  'Flat
         BackColor       =   &H00C0C0C0&
         BorderStyle     =   0  'None
         FillColor       =   &H0000FFFF&
         FillStyle       =   0  'Solid
         ForeColor       =   &H80000008&
         Height          =   3855
         Left            =   60
         ScaleHeight     =   80
         ScaleMode       =   0  'User
         ScaleWidth      =   80
         TabIndex        =   25
         Top             =   60
         Width           =   3855
         Begin VB.Line zHLine
            BorderWidth     =   6
            Index           =   0
            X1              =   0
            X2              =   80
            Y1              =   40.01
            Y2              =   40.01
         End
         Begin VB.Line zVLine
            BorderWidth     =   6
            Index           =   0
            X1              =   40.01
            X2              =   40.01
            Y1              =   0
            Y2              =   80
         End
         Begin VB.Line zHLine
            BorderWidth     =   3
            Index           =   1
            X1              =   0
            X2              =   80
            Y1              =   20.005
            Y2              =   20.005
         End
         Begin VB.Line zHLine
            BorderWidth     =   3
            Index           =   2
            X1              =   0
            X2              =   80
            Y1              =   59.995
            Y2              =   59.995
         End
         Begin VB.Line zHLine
            Index           =   3
            X1              =   0
            X2              =   80
            Y1              =   69.997
```

```
            Y2              =   69.997
         End
         Begin VB.Line zHLine
            Index           =   4
            X1              =   0
            X2              =   80
            Y1              =   49.992
            Y2              =   49.992
         End
         Begin VB.Line zHLine
            Index           =   5
            X1              =   0
            X2              =   80
            Y1              =   30.008
            Y2              =   30.008
         End
         Begin VB.Line zHLine
            Index           =   6
            X1              =   0
            X2              =   80
            Y1              =   10.003
            Y2              =   10.003
         End
         Begin VB.Line zVLine
            BorderWidth     =   3
            Index           =   1
            X1              =   59.995
            X2              =   59.995
            Y1              =   0
            Y2              =   80
         End
         Begin VB.Line zVLine
            BorderWidth     =   3
            Index           =   2
            X1              =   20.005
            X2              =   20.005
            Y1              =   0
            Y2              =   80
         End
         Begin VB.Line zVLine
            Index           =   3
            X1              =   10.003
            X2              =   10.003
            Y1              =   0
            Y2              =   80
         End
         Begin VB.Line zVLine
            Index           =   4
            X1              =   30.008
            X2              =   30.008
            Y1              =   0
            Y2              =   80
         End
         Begin VB.Line zVLine
            Index           =   5
            X1              =   49.992
            X2              =   49.992
            Y1              =   0
            Y2              =   80
         End
         Begin VB.Line zVLine
            Index           =   6
            X1              =   69.997
            X2              =   69.997
            Y1              =   0
            Y2              =   80
         End
      End
   End
```

```
      Begin Threed.SSOption optNE
         Height          =   195
         Left            =   4400
         TabIndex        =   33
         Top             =   480
         Width           =   195
         _version        =   65536
         _extentx        =   344
         _extenty        =   344
         _stockprops     =   78
         caption         =   "NE"
      End
      Begin Threed.SSOption optNW
         Height          =   255
         Left            =   120
         TabIndex        =   32
         Top             =   480
         Width           =   227
         _version        =   65536
         _extentx        =   397
         _extenty        =   450
         _stockprops     =   78
         caption         =   "Option3D3"
      End
      Begin Threed.SSOption optSE
         Height          =   255
         Left            =   4400
         TabIndex        =   31
         Top             =   4170
         Width           =   255
         _version        =   65536
         _extentx        =   450
         _extenty        =   450
         _stockprops     =   78
         caption         =   "Option3D2"
      End
      Begin Threed.SSOption optSW
         Height          =   255
         Left            =   120
         TabIndex        =   30
         Top             =   4170
         Width           =   227
         _version        =   65536
         _extentx        =   397
         _extenty        =   450
         _stockprops     =   78
         caption         =   "Option3D1"
      End
      Begin VB.Label zLabelY
         Appearance      =   0  'Flat
         BackColor       =   &H80000005&
         BackStyle       =   0  'Transparent
         Caption         =   "&X"
         ForeColor       =   &H80000008&
         Height          =   255
         Left            =   4560
         TabIndex        =   50
         Top             =   3300
         Width           =   255
      End
      Begin VB.Label zLabelX
         Appearance      =   0  'Flat
         BackColor       =   &H80000005&
         BackStyle       =   0  'Transparent
         Caption         =   "&Y"
         ForeColor       =   &H80000008&
         Height          =   255
         Left            =   6540
         TabIndex        =   49
```

```
      Top             =   3300
      Width           =   195
   End
   Begin VB.Label zlblSection
      Appearance      =   0  'Flat
      BackColor       =   &H80000005&
      BackStyle       =   0  'Transparent
      Caption         =   "Section Name as TTdRRdSS"
      BeginProperty Font
         name            =   "MS Sans Serif"
         charset         =   1
         weight          =   400
         size            =   8.25
         underline       =   0   'False
         italic          =   0   'False
         strikethrough   =   0   'False
      EndProperty
      ForeColor       =   &H80000008&
      Height          =   255
      Left            =   4740
      TabIndex        =   48
      Top             =   420
      Width           =   2235
   End
End
Begin Threed.SSPanel pnlQTSubdivision
   Height          =   4680
   Left            =   150
   TabIndex        =   6
   Top             =   750
   Width           =   8700
   _version        =   65536
   _extentx        =   15346
   _extenty        =   8255
   _stockprops     =   15
   bevelwidth      =   2
   outline         =   -1  'True
   Begin VB.ComboBox cmbAllFields
      Appearance      =   0  'Flat
      Height          =   300
      Left            =   2050
      Style           =   2  'Dropdown List
      TabIndex        =   59
      Top             =   480
      Width           =   6550
   End
   Begin VB.CommandButton cmdStartConversion
      Appearance      =   0  'Flat
      BackColor       =   &H80000005&
      Caption         =   "Convert &Legal Descriptions to XY Pairs"
      Height          =   315
      Left            =   4440
      TabIndex        =   0
      Top             =   3480
      Width           =   3975
   End
   Begin VB.TextBox txtYFieldOut
      Appearance      =   0  'Flat
      Height          =   300
      Left            =   1740
      TabIndex        =   1
      Text            =   "Y_FromLegalDescription"
      Top             =   3840
      Width           =   2535
   End
   Begin VB.TextBox txtXFieldOut
      Appearance      =   0  'Flat
      Height          =   300
      Left            =   1740
```

```
      TabIndex        =   2
      Text            =   "X_FromLegalDescription"
      Top             =   3480
      Width           =   2535
   End
   Begin Threed.SSFrame fraFootage
      Height          =   1830
      Left            =   4380
      TabIndex        =   57
      Top             =   1545
      Width           =   4095
      _version        =   65536
      _extentx        =   7223
      _extenty        =   3228
      _stockprops     =   14
      caption         =   " Offsets  "
      forecolor       =   0
      BeginProperty font {FB8F0823-0164-101B-84ED-08002B2EC713}
         name            =   "MS Sans Serif"
         charset         =   1
         weight          =   400
         size            =   8.25
         underline       =   0   'False
         italic          =   0   'False
         strikethrough   =   0   'False
      EndProperty
      Begin VB.TextBox txtUnitFactor
         Appearance      =   0  'Flat
         BeginProperty Font
            name            =   "MS Sans Serif"
            charset         =   1
            weight          =   400
            size            =   8.25
            underline       =   0   'False
            italic          =   0   'False
            strikethrough   =   0   'False
         EndProperty
         Height          =   300
         Left            =   120
         TabIndex        =   3
         Text            =   "0.3048"
         Top             =   1440
         Width           =   795
      End
      Begin VB.PictureBox picGray
         Appearance      =   0  'Flat
         BackColor       =   &H00C0C0C0&
         BorderStyle     =   0  'None
         ForeColor       =   &H80000008&
         Height          =   300
         Index           =   6
         Left            =   2595
         ScaleHeight     =   300
         ScaleWidth      =   105
         TabIndex        =   4
         Top             =   1080
         Width           =   100
      End
      Begin VB.ComboBox cmbNSOffset
         Appearance      =   0  'Flat
         Height          =   300
         Left            =   120
         TabIndex        =   5
         Text            =   "0"
         Top             =   1080
         Width           =   2835
      End
      Begin VB.PictureBox picGray
         Appearance      =   0  'Flat
```

```
      BackColor       =   &H00C0C0C0&
      BorderStyle     =   0  'None
      ForeColor       =   &H80000008&
      Height          =   300
      Index           =   5
      Left            =   2595
      ScaleHeight     =   300
      ScaleWidth      =   105
      TabIndex        =   11
      Top             =   720
      Width           =   100
   End
   Begin VB.ComboBox cmbEWOffset
      Appearance      =   0  'Flat
      Height          =   300
      Left            =   120
      TabIndex        =   14
      Text            =   "0"
      Top             =   720
      Width           =   2835
   End
   Begin VB.PictureBox picGray
      Appearance      =   0  'Flat
      BackColor       =   &H00C0C0C0&
      BorderStyle     =   0  'None
      ForeColor       =   &H80000008&
      Height          =   300
      Index           =   4
      Left            =   2595
      ScaleHeight     =   300
      ScaleWidth      =   105
      TabIndex        =   15
      Top             =   360
      Width           =   100
   End
   Begin VB.ComboBox cmbFromCorner
      Appearance      =   0  'Flat
      Height          =   300
      Left            =   120
      TabIndex        =   16
      Text            =   """SW"""
      Top             =   360
      Width           =   2835
   End
   Begin VB.Label zlblUnitMultiplier
      Appearance      =   0  'Flat
      BackColor       =   &H80000005&
      BackStyle       =   0  'Transparent
      Caption         =   "Unit Multiplier from Data Units to Offsets"
      BeginProperty Font
         name            =   "MS Sans Serif"
         charset         =   1
         weight          =   400
         size            =   8.25
         underline       =   0   'False
         italic          =   0   'False
         strikethrough   =   0   'False
      EndProperty
      ForeColor       =   &H80000008&
      Height          =   255
      Index           =   1
      Left            =   960
      TabIndex        =   17
      Top             =   1440
      Width           =   3015
   End
   Begin VB.Label Label1
      Appearance      =   0  'Flat
      BackColor       =   &H80000005&
```

```
      BackStyle       =   0  'Transparent
      Caption         =   "N-S Offset"
      BeginProperty Font
         name            =   "MS Sans Serif"
         charset         =   1
         weight          =   400
         size            =   8.25
         underline       =   0   'False
         italic          =   0   'False
         strikethrough   =   0   'False
      EndProperty
      ForeColor       =   &H80000008&
      Height          =   255
      Left            =   3000
      TabIndex        =   18
      Top             =   1080
      Width           =   975
   End
   Begin VB.Label zlblEWOffset
      Appearance      =   0  'Flat
      BackColor       =   &H80000005&
      BackStyle       =   0  'Transparent
      Caption         =   "E-W Offset"
      BeginProperty Font
         name            =   "MS Sans Serif"
         charset         =   1
         weight          =   400
         size            =   8.25
         underline       =   0   'False
         italic          =   0   'False
         strikethrough   =   0   'False
      EndProperty
      ForeColor       =   &H80000008&
      Height          =   255
      Left            =   3000
      TabIndex        =   19
      Top             =   720
      Width           =   1035
   End
   Begin VB.Label zlblCorner
      Appearance      =   0  'Flat
      BackColor       =   &H80000005&
      BackStyle       =   0  'Transparent
      Caption         =   "from Corner"
      BeginProperty Font
         name            =   "MS Sans Serif"
         charset         =   1
         weight          =   400
         size            =   8.25
         underline       =   0   'False
         italic          =   0   'False
         strikethrough   =   0   'False
      EndProperty
      ForeColor       =   &H80000008&
      Height          =   255
      Left            =   3000
      TabIndex        =   20
      Top             =   360
      Width           =   1035
   End
End
Begin Threed.SSFrame fraSubsection
   Height          =   1830
   Left            =   180
   TabIndex        =   51
   Top             =   1545
   Width           =   4095
   _version        =   65536
   _extentx        =   7223
```

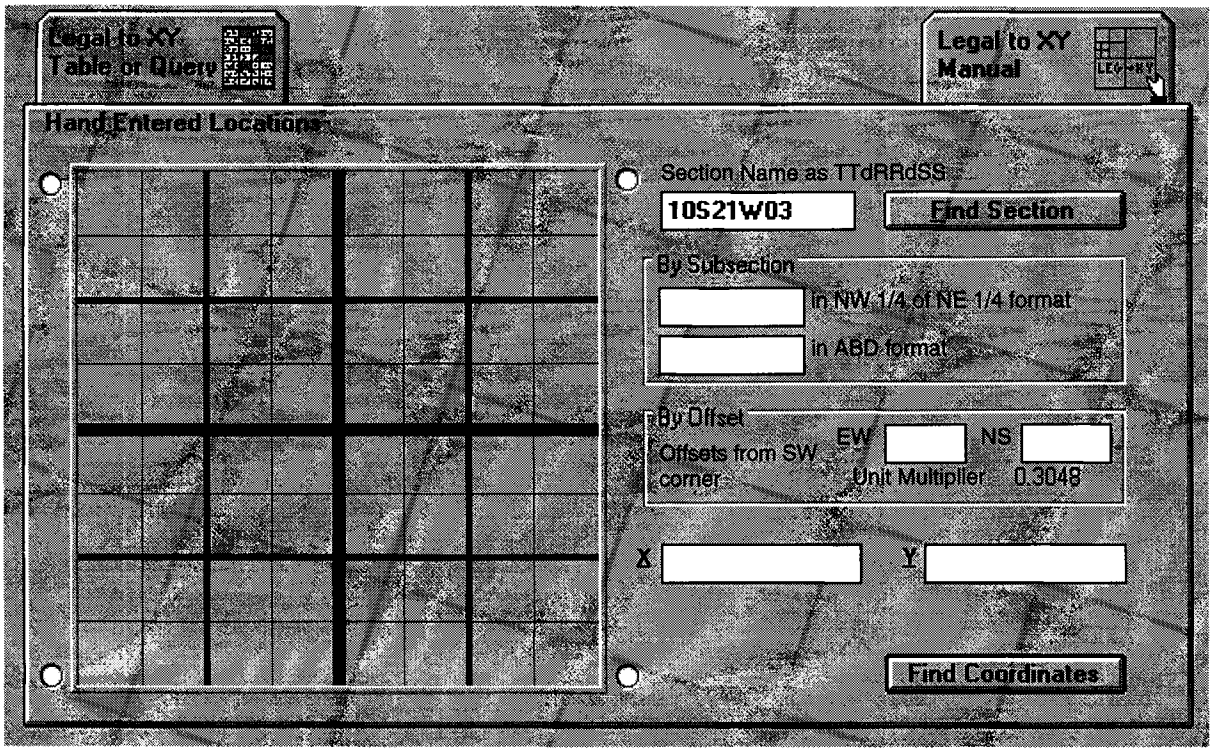
```
      _extenty        =   3228
      _stockprops     =   14
      caption         =   " Subsection  "
      forecolor       =   0
      BeginProperty font {FB8F0823-0164-101B-84ED-08002B2EC713}
         name            =   "MS Sans Serif"
         charset         =   1
         weight          =   400
         size            =   8.25
         underline       =   0   'False
         italic          =   0   'False
         strikethrough   =   0   'False
      EndProperty
      Begin VB.ComboBox cmbABCD
         Appearance      =   0  'Flat
         Height          =   300
         Left            =   120
         Style           =   2  'Dropdown List
         TabIndex        =   58
         Top             =   720
         Width           =   2835
      End
      Begin VB.ComboBox cmbNWofNE
         Appearance      =   0  'Flat
         Height          =   300
         Left            =   120
         Style           =   2  'Dropdown List
         TabIndex        =   56
         Top             =   300
         Width           =   2835
      End
      Begin Threed.SSOption optAsABCD
         Height          =   195
         Left            =   3000
         TabIndex        =   55
         Top             =   780
         Width           =   915
         _version        =   65536
         _extentx        =   1614
         _extenty        =   344
         _stockprops     =   78
         caption         =   "as ADCB"
         BeginProperty font {FB8F0823-0164-101B-84ED-08002B2EC713}
            name            =   "MS Sans Serif"
            charset         =   1
            weight          =   400
            size            =   8.25
            underline       =   0   'False
            italic          =   0   'False
            strikethrough   =   0   'False
         EndProperty
      End
      Begin Threed.SSOption optAsNWofNE
         Height          =   255
         Left            =   3000
         TabIndex        =   54
         Top             =   300
         Width           =   1035
         _version        =   65536
         _extentx        =   1826
         _extenty        =   450
         _stockprops     =   78
         caption         =   "as NW1/4"
         BeginProperty font {FB8F0823-0164-101B-84ED-08002B2EC713}
            name            =   "MS Sans Serif"
            charset         =   1
            weight          =   400
            size            =   8.25
            underline       =   0   'False
```

```
            italic          =   0   'False
            strikethrough   =   0   'False
         EndProperty
      End
   End
   Begin VB.PictureBox picGray
      Appearance      =   0  'Flat
      BackColor       =   &H00C0C0C0&
      BorderStyle     =   0  'None
      ForeColor       =   &H80000008&
      Height          =   300
      Index           =   1
      Left            =   8120
      ScaleHeight     =   300
      ScaleWidth      =   105
      TabIndex        =   13
      Top             =   1680
      Width           =   100
   End
   Begin VB.Data dat_SectionConversion
      Appearance      =   0  'Flat
      Caption         =   "Move record "
      Connect         =   ""
      DatabaseName    =   ""
      Exclusive       =   0   'False
      Height          =   270
      Left            =   9120
      Options         =   0
      ReadOnly        =   0   'False
      RecordsetType   =   1  'Dynaset
      RecordSource    =   ""
      Top             =   1200
      Width           =   2445
   End
   Begin VB.ComboBox cmbQTList_SubSection
      Appearance      =   0  'Flat
      Height          =   300
      Left            =   90
      Sorted          =   -1  'True
      Style           =   2  'Dropdown List
      TabIndex        =   10
      Top             =   120
      Width           =   8500
   End
   Begin VB.ComboBox cmbSectionName
      Appearance      =   0  'Flat
      Height          =   300
      Left            =   4080
      Style           =   2  'Dropdown List
      TabIndex        =   9
      Top             =   950
      Width           =   4515
   End
   Begin VB.PictureBox picPercentDoneHost
      Appearance      =   0  'Flat
      BackColor       =   &H80000005&
      ForeColor       =   &H80000008&
      Height          =   255
      Left            =   120
      ScaleHeight     =   1
      ScaleMode       =   0  'User
      ScaleWidth      =   96.546
      TabIndex        =   7
      Top             =   4320
      Width           =   8415
      Begin VB.PictureBox picPercentDone
         Appearance      =   0  'Flat
         BackColor       =   &H00FF0000&
         BorderStyle     =   0  'None
```

```
      ForeColor       =   &H80000008&
      Height          =   225
      Left            =   0
      ScaleHeight     =   225
      ScaleWidth      =   15
      TabIndex        =   8
      Top             =   0
      Width           =   15
   End
End
Begin Threed.SSOption optFootage
   Height          =   195
   Left            =   4380
   TabIndex        =   53
   Top             =   1335
   Width           =   2655
   _version        =   65536
   _extentx        =   4683
   _extenty        =   344
   _stockprops     =   78
   caption         =   "Point locations by &Footages"
   BeginProperty font {FB8F0823-0164-101B-84ED-08002B2EC713}
      name            =   "MS Sans Serif"
      charset         =   1
      weight          =   400
      size            =   8.25
      underline       =   0   'False
      italic          =   0   'False
      strikethrough   =   0   'False
   EndProperty
End
Begin Threed.SSOption optSubSection
   Height          =   255
   Left            =   180
   TabIndex        =   52
   Top             =   1335
   Width           =   4035
   _version        =   65536
   _extentx        =   7117
   _extenty        =   450
   _stockprops     =   78
   caption         =   "Point locations by Subdividing or &Quartering"
   BeginProperty font {FB8F0823-0164-101B-84ED-08002B2EC713}
      name            =   "MS Sans Serif"
      charset         =   1
      weight          =   400
      size            =   8.25
      underline       =   0   'False
      italic          =   0   'False
      strikethrough   =   0   'False
   EndProperty
End
Begin MSComDlg.CommonDialog CMDialog1
   Left            =   9120
   Top             =   4320
   _version        =   65536
   _extentx        =   847
   _extenty        =   847
   _stockprops     =   0
End
Begin VB.Label Label2
   Appearance      =   0  'Flat
   BackColor       =   &H80000005&
   BackStyle       =   0  'Transparent
   Caption         =   "&All Fields in Table or Query"
   BeginProperty Font
      name            =   "MS Sans Serif"
      charset         =   1
      weight          =   400
```

```
         size            =   8.25
         underline       =   0   'False
         italic          =   0   'False
         strikethrough   =   0   'False
      EndProperty
      ForeColor       =   &H80000008&
      Height          =   255
      Left            =   120
      TabIndex        =   60
      Top             =   480
      Width           =   2055
   End
   Begin VB.Label zlblY
      Appearance      =   0  'Flat
      BackColor       =   &H80000005&
      BackStyle       =   0  'Transparent
      Caption         =   "Output &Y Field Name"
      BeginProperty Font
         name            =   "MS Sans Serif"
         charset         =   1
         weight          =   400
         size            =   8.25
         underline       =   0   'False
         italic          =   0   'False
         strikethrough   =   0   'False
      EndProperty
      ForeColor       =   &H80000008&
      Height          =   255
      Left            =   180
      TabIndex        =   21
      Top             =   3840
      Width           =   1575
   End
   Begin VB.Label zlblX
      Appearance      =   0  'Flat
      BackColor       =   &H80000005&
      BackStyle       =   0  'Transparent
      Caption         =   "Output &X Field Name"
      BeginProperty Font
         name            =   "MS Sans Serif"
         charset         =   1
         weight          =   400
         size            =   8.25
         underline       =   0   'False
         italic          =   0   'False
         strikethrough   =   0   'False
      EndProperty
      ForeColor       =   &H80000008&
      Height          =   255
      Left            =   180
      TabIndex        =   22
      Top             =   3480
      Width           =   1635
   End
   Begin VB.Label zlblSectionsFrom
      Appearance      =   0  'Flat
      BackColor       =   &H80000005&
      BackStyle       =   0  'Transparent
      Caption         =   "Field containing section name as TTdRRdSS"
      ForeColor       =   &H80000008&
      Height          =   255
      Left            =   240
      TabIndex        =   12
      Top             =   950
      Width           =   3975
   End
End
Begin Threed.SSRibbon cmdManual
   Height          =   720
```

```
      Left            =   6825
      TabIndex        =   61
      Top             =   75
      Width           =   1920
      _version        =   65536
      _extentx        =   3387
      _extenty        =   1270
      _stockprops     =   65
      value           =   -1  'True
      autosize        =   1
      bevelwidth      =   0
      outline         =   0   'False
      pictureup       =   {Binary}
      picturedn       =   {Binary}
   End
   Begin Threed.SSRibbon cmdFromDB
      Height          =   720
      Left            =   240
      TabIndex        =   62
      Top             =   75
      Width           =   1920
      _version        =   65536
      _extentx        =   3387
      _extenty        =   1270
      _stockprops     =   65
      groupallowallup =   -1  'True
      autosize        =   1
      bevelwidth      =   0
      outline         =   0   'False
      pictureup       =   {Binary}
      picturedn       =   {Binary}
   End
   Begin VB.Menu zzzzFile
      Caption         =   "&File"
      Begin VB.Menu mfOpenDB
         Caption         =   "&Open Database..."
      End
      Begin VB.Menu mfCloseDatabase
         Caption         =   "&Close Database"
      End
      Begin VB.Menu zzzzz1
         Caption         =   "-"
      End
      Begin VB.Menu mfExit
         Caption         =   "E&xit"
      End
   End
   Begin VB.Menu zzMnuEdit
      Caption         =   "&Edit"
      Begin VB.Menu meCopy
         Caption         =   "&Copy"
         Shortcut        =   ^{INSERT}
      End
      Begin VB.Menu mePaste
         Caption         =   "&Paste"
         Shortcut        =   +{INSERT}
      End
   End
End
```


Legal to XY
Table or Query
Legal to XY
Manual
Hand Entered Locations
Section Name as TTdRRdSS
10S21W03
Find Section
By Subsection
In NW 1/4 of NE 1/4 format
In ABD format
By Offset
Offsets from SW corner
EW
NS
Unit Multiplier 0.3048
X
Y
Find Coordinates

KSSAS004 - 1

```
Option Explicit
Dim sngTemp1 As Single, sngTemp2 As Single
Dim intTemp3 As Integer, intTemp4 As Integer
Dim lngTemp5 As Long, lngTemp6 As Long
Dim varTemp7 As Variant, varTemp8 As Variant

Type typIntXY
    X As Integer
    Y As Integer
End Type
Type typLong
    a As Long
End Type




Global Const SECTION_STEP = 32
'The SECTION_STEP=32 is used to allow subdivision into smaller areas and still use short intege
rs.
' For use of the quartering options, it must be a power of two.
' If SECTION_STEP=2**N, you can have N-1 levels of halving/quartering and still have an integer
Const OFFSET_RANGE = 44, OFFSET_EAST = 600
'  OFFSET_RANGE specifies that the counting starts from Range 44W
'  OFFSET_EAST specifies the false easting (in miles) added to the baseline (at 44W)
'  Note that OFFSET_EAST  MUST be a multiple of 6 for the inverse routines to work.
Const OFFSET_TIER = 36, OFFSET_NORTH = 60
' specifies that T36S will have false northing of 60 miles
'  Note that OFFSET_NORTH  MUST be a multiple of 6 for the inverse routines to work.
'    (northings in 1/32th miles become 4 digit numbers for whole state)


'All of the functions in this module are explicitly VARIANT, so that they
'  can return error/null values to keep typos from propogating







'Inverse Transormations

'^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
'------------------------------------------------------------------------------
'vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
 Function kssas_ColumnFromRangeName(ByVal RangeName As String) As Variant
''Converts Range name like 01E 43W 025E  to the Column number.
''  Range 44W
''(which does not exist, but would be south of the state) is row 95
''GWPouch 95 03 03
 RangeName = UCase$(Trim$(RangeName))
 If Len(RangeName) <> 3 Then
        kssas_ColumnFromRangeName = Null    '  for most VBasics  CVErr(xlErrNA)  for Excel
        Exit Function
 End If
 Dim RangeNum As Integer
 Select Case Right$(RangeName, 1)
    Case "W"
      RangeNum = OFFSET_RANGE - Val(Left$(RangeName, 2))
    Case "E"
      RangeNum = OFFSET_RANGE - 1 + Val(Left$(RangeName, 2))
    Case Else
         kssas_ColumnFromRangeName = Null    '  for most VBasics  CVErr(xlErrNA)  for Excel
         Exit Function
 End Select
```

```
 kssas_ColumnFromRangeName = SECTION_STEP * (RangeNum * 6 + OFFSET_EAST)
End Function

Function kssas_ColumnFromSectionNum(ByVal SectNum As Integer) As Variant
'' Converts section number (1 to 36) to the column number.
''  Western most column is #0, easternmost column is number #5
''GWPouch 95 03 03
'This function is of type Variant, so that I can return an error value
    If SectNum < 1 Or SectNum > 36 Then
        kssas_ColumnFromSectionNum = Null    '  for most VBasics  CVErr(xlErrNA)  for Excel
        Exit Function
    End If
    intTemp3 = (SectNum - 1) \ 6
    If intTemp3 Mod 2 Then
        kssas_ColumnFromSectionNum = SECTION_STEP * (0 + ((SectNum - 1) Mod 6))
      Else
        kssas_ColumnFromSectionNum = SECTION_STEP * (5 - ((SectNum - 1) Mod 6))
    End If
End Function

'^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
'------------------------------------------------------------------------------
'vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
'Offsets for subsections
Function kssas_ColumnFromSubSectABCD(ByVal SubSect As String) As Variant
'' This function returns the kssas column offset
''  from a BLM/USGS style ABBDC style subsection.  Add to section columns

'' Greg Pouch  KGS 95 03 07
SubSect = UCase$(Trim$(SubSect))

Dim CurCol As Single, CurPower As Integer, n As Long, n2 As Long
CurCol = SECTION_STEP \ 2 'Start in the middle
For n = 1 To Len(SubSect)
    Select Case Mid$(SubSect, n, 1)
       Case "A", "D" 'in the east half, so add
          CurCol = CurCol + SECTION_STEP / 2 ^ (n + 1)
       Case "B", "C" 'in west half, so subtract
          CurCol = CurCol - SECTION_STEP / 2 ^ (n + 1)
       Case Else
          kssas_ColumnFromSubSectABCD = Null    '  for most VBasics  CVErr(xlErrNA)  for Excel
          Exit Function
    End Select
Next n
kssas_ColumnFromSubSectABCD = CurCol
End Function

Function kssas_ColumnFromSubsectSEofNW(ByVal SubSect As String) As Variant
'' Returns a kssas column offset from a small to big  NE1/4 of NW1/4 of .... style subsection
'' Greg Pouch  KGS 95 03 07   Calling routine must trim the string, PROPERLY
SubSect = UCase$((SubSect))

Dim CurCol As Single, CurPower As Integer, n As Long, n2 As Long
CurCol = SECTION_STEP \ 2 'Start in the middle
CurPower = Len(SubSect) / 2 + 1
For n = 1 To Len(SubSect) Step 2
    Select Case Mid$(SubSect, n + 1, 1)
       Case "E" 'in the east half, so add
          CurCol = CurCol + SECTION_STEP / 2 ^ CurPower
       Case "W" 'in west half, so subtract
          CurCol = CurCol - SECTION_STEP / 2 ^ CurPower
       Case " ", "_", "C", "X"

       Case Else
          kssas_ColumnFromSubsectSEofNW = Null    '  for most VBasics  CVErr(xlErrNA)  for Excel
          Exit Function
    End Select
    CurPower = CurPower - 1
Next n
```

```
kssas_ColumnFromSubsectSEofNW = CurCol
End Function

'^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
'------------------------------------------------------------------------------
'vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
Function kssas_ColumnFromTTdRRdSS(ByVal TTdRRdSS As String) As Variant
''Converts Section name like 22S33W01 to the Column number.
''GWPouch 95 03 03

 TTdRRdSS = UCase$(Trim$(TTdRRdSS))
 If Len(TTdRRdSS) <> 8 Then
        kssas_ColumnFromTTdRRdSS = Null    '  for most VBasics  CVErr(xlErrNA)  for Excel
        Exit Function
 End If

 Dim RangeName As String, SectNum As Integer
 RangeName = Mid$(TTdRRdSS, 4, 3)
 SectNum = CInt(Right$(TTdRRdSS, 2))
 kssas_ColumnFromTTdRRdSS = kssas_ColumnFromRangeName(RangeName) + kssas_ColumnFromSectionNum(S
ectNum)
End Function

Function kssas_GetSECTION_STEP()
  kssas_GetSECTION_STEP = SECTION_STEP
End Function

Function kssas_RangeFromColumn(ByVal ColNum As Integer) As Variant
'' Returns the range in nnD format (like 35W or 01E) from a kssas column number
'' Greg Pouch  KGS 95 03 07

'This odd looking constant is the principal meridian
 If ColNum < 27648 Then
   'West of 6th Principal Meridian
   kssas_RangeFromColumn = Format(OFFSET_RANGE - (ColNum \ SECTION_STEP - OFFSET_EAST) \ 6, "00
") & "W"
  Else
   kssas_RangeFromColumn = Format((ColNum \ SECTION_STEP - OFFSET_EAST) \ 6 - OFFSET_RANGE + 1,
 "00") & "E"
 End If
                    '=IF(kssas_Easting<20256) then
                        '   TEXT(43-INT((kssas_Easting-12000)/192),"00")&"W"
                        '   Else
                    '    TEXT(INT((kssas_Easting-20064)/192),"00")&"E")
End Function

'^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
'------------------------------------------------------------------------------
'vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
 Function kssas_RowFromSectionNum(ByVal SectNum As Integer) As Variant
''Converts section number (1 to 36) to the row number.
''  Southern most row is #0,northernmost row is number #5
''GWPouch 95 03 03
 'This function is of type Variant, so that I can return an error value
    If SectNum < 1 Or SectNum > 36 Then
        kssas_RowFromSectionNum = Null    '  for most VBasics  CVErr(xlErrNA)  for Excel
        Exit Function
    End If
    kssas_RowFromSectionNum = SECTION_STEP * (5 - (SectNum - 1) \ 6)
End Function

Function kssas_RowFromSubSectABCD(ByVal SubSect As String) As Variant
'' Returns a kssas Row offset from a USGS/BLM style aadcb style subsection

'' Greg Pouch  KGS 95 03 07
SubSect = UCase$(Trim$(SubSect))

Dim CurRow As Single, n As Long
CurRow = SECTION_STEP \ 2 'Start in the middle
```

```
For n = 1 To Len(SubSect)
    Select Case Mid$(SubSect, n, 1)
       Case "A", "B" 'in the north half, so add
          CurRow = CurRow + SECTION_STEP / 2 ^ (n + 1)
       Case "C", "D" 'in soth half, so subtract
          CurRow = CurRow - SECTION_STEP / 2 ^ (n + 1)
       Case Else
          kssas_RowFromSubSectABCD = Null    '  for most VBasics  CVErr(xlErrNA)  for Excel ' Nu
ll  '  for most VBasics  CVErr(xlErrNA)  for Excel
          Exit Function
    End Select
Next n
kssas_RowFromSubSectABCD = CurRow
End Function

Function kssas_RowFromSubsectSEofNW(ByVal SubSect As String) As Variant
'' Returns a kssas row offset from a small to big  NE1/4 of NW1/4 of .... style subsection
'' Greg Pouch  KGS 95 03 07



SubSect = UCase$((SubSect))

Dim CurRow As Single, CurPower As Integer, n As Long, n2 As Long
CurRow = SECTION_STEP \ 2 'Start in the middle
CurPower = Len(SubSect) / 2 + 1
For n = 1 To Len(SubSect) Step 2
    Select Case Mid$(SubSect, n, 1)
       Case "N" 'in the north half, so add
          CurRow = CurRow + SECTION_STEP / 2 ^ CurPower
       Case "S" 'in south half, so subtract
          CurRow = CurRow - SECTION_STEP / 2 ^ CurPower
       Case " ", "_", "C", "X"
       Case Else
          kssas_RowFromSubsectSEofNW = Null    '  for most VBasics  CVErr(xlErrNA)  for Excel
          Exit Function
    End Select
    CurPower = CurPower - 1
Next n
kssas_RowFromSubsectSEofNW = CurRow
End Function

'^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
'------------------------------------------------------------------------------
'vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
Function kssas_RowFromTierName(ByVal TierName As String)
''Converts Tier name like 01S  03S 35S 10N to the row number.
''  Tier 36S (which does not exist, but would be south of the state) is row 95
''GWPouch 95 03 03
 TierName = UCase$(Trim$(TierName))
 If Len(TierName) <> 3 Then
        kssas_RowFromTierName = Null    '  for most VBasics  CVErr(xlErrNA)  for Excel
        Exit Function
 End If
 Dim TierNum As Integer
 TierNum = Val(Left$(TierName, 2))
 Select Case Right$(TierName, 1)
     Case "N"
      TierNum = 1 - TierNum
     Case "S"
      'Do Nothing
     Case Else
       kssas_RowFromTierName = Null    '  for most VBasics  CVErr(xlErrNA)  for Excel
       Exit Function
 End Select
 kssas_RowFromTierName = SECTION_STEP * (OFFSET_NORTH + (OFFSET_TIER - TierNum) * 6)
End Function

'^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

```
'-------------------------------------------------------------------------
'vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
 Function kssas_RowFromTierNumS(ByVal TierNum As Integer) As Variant
''Converts Tier number (1 to 35) to the row number.
''  Tier 36S (which does not exist) is row 95
''GWPouch 95 03 03
 If TierNum < 1 Or TierNum > 35 Then
        kssas_RowFromTierNumS = Null    '  for most VBasics  CVErr(xlErrNA)  for Excel
        Exit Function
 End If
 kssas_RowFromTierNumS = SECTION_STEP * (OFFSET_NORTH + (OFFSET_TIER - TierNum) * 6)
End Function

'^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
'-------------------------------------------------------------------------
'vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
Function kssas_RowFromTTdRRdSS(ByVal TTdRRdSS As String) As Variant
''Converts Section name like 22S33W01 to the row number.
''GWPouch 95 03 03

 TTdRRdSS = UCase$(Trim$(TTdRRdSS))
 If Len(TTdRRdSS) <> 8 Then
        kssas_RowFromTTdRRdSS = Null    '  for most VBasics  CVErr(xlErrNA)  for Excel
        Exit Function
 End If

 Dim TierName As String, SectNum As Integer
 TierName = Mid$(TTdRRdSS, 1, 3)
 SectNum = CInt(Right$(TTdRRdSS, 2))
 kssas_RowFromTTdRRdSS = kssas_RowFromTierName(TierName) + kssas_RowFromSectionNum(SectNum)
End Function

'^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
'-------------------------------------------------------------------------
'vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
Function kssas_SectionFromColumnRow(ByVal ColNum As Integer, ByVal RowNum As Integer) As Varian
t
'' Returns the section in  from a kssas column and row pair
'' Greg Pouch  KGS 95 03 07


Dim ColOffset As Integer, RowOffset As Integer, Column As Integer, Row As Integer
'First, get the column number for the whole section.
    Column = ColNum \ SECTION_STEP
'Next, remove the offset for the Range
    ColOffset = ((ColNum \ SECTION_STEP - OFFSET_EAST) \ 6) * 6 + OFFSET_EAST
    Column = Column - ColOffset
'Get the row number for the whole section
    Row = RowNum \ SECTION_STEP
'Next, remove the offset for the Tier
    RowOffset = ((RowNum \ SECTION_STEP - OFFSET_NORTH) \ 6) * 6 + OFFSET_NORTH
    Row = Row - RowOffset

    If Row Mod 2 = 0 Then
        'The southern row of a township is 0.
        kssas_SectionFromColumnRow = (5 - Row) * 6 + 1 + Column
      Else
        kssas_SectionFromColumnRow = (6 - Row) * 6 - Column
    End If
                    'IF(MOD(INT(MOD(S_kssas-2400,192)/32),2),7+INT(MOD(E_kssas-12000,192)/32)+6
*(INT(MOD(S_kssas-2400,192)/32)-1),6-INT(MOD(E_kssas-12000,192)/32)+6*INT(MOD(S_kssas-2400,192)
/32))
End Function

'^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
'-------------------------------------------------------------------------
'vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
Function kssas_SerialNumberFromTTdRRdSS(ByVal TTdRRdSS As String) As Variant
''This function takes a simple Tier Range Section description
```

```
'' and converts it to a 4 Byte serial number
'' Greg Pouch  KGS 95 03 07
   Dim IntXY As typIntXY, LngNum As typLong

    TTdRRdSS = UCase$(Trim$(TTdRRdSS))

    varTemp7 = kssas_ColumnFromTTdRRdSS(TTdRRdSS)
    varTemp8 = kssas_RowFromTTdRRdSS(TTdRRdSS)

    If (IsNull(varTemp7) Or IsNull(varTemp8)) Then
          kssas_SerialNumberFromTTdRRdSS = Null    '  for most VBasics  CVErr(xlErrNA)  for Exce
l
        Else
          IntXY.X = varTemp7
          IntXY.Y = varTemp8
          LSet LngNum = IntXY
          kssas_SerialNumberFromTTdRRdSS = LngNum.a
    End If
End Function

Function kssas_SerialNumberOffsetFromABCD(ByVal SubSect As String) As Variant
'' Returns a kssas serial number offset from a BLM/USGS style subsection abdca
'' Greg Pouch  KGS 95 03 07


    varTemp7 = kssas_ColumnFromSubSectABCD(SubSect)
    varTemp8 = kssas_RowFromSubSectABCD(SubSect)
Dim IntXY As typIntXY, LngNum As typLong
    If (IsNull(varTemp7) Or IsNull(varTemp8)) Then
          kssas_SerialNumberOffsetFromABCD = Null    '  for most VBasics  CVErr(xlErrNA)  for Ex
cel
        Else
          IntXY.X = varTemp7
          IntXY.Y = varTemp8
          LSet LngNum = IntXY
          kssas_SerialNumberOffsetFromABCD = LngNum.a
    End If
End Function

Function kssas_SerialNumberOffsetFromSEofNW(ByVal SubSect As String) As Variant
'' Returns a kssas serial number from a small to big  NE1/4 of NW1/4 of .... style subsection
'' Greg Pouch  KGS 95 03 07


    varTemp7 = kssas_ColumnFromSubsectSEofNW(SubSect)
    varTemp8 = kssas_RowFromSubsectSEofNW(SubSect)
Dim IntXY As typIntXY, LngNum As typLong
    If (IsNull(varTemp7) Or IsNull(varTemp8)) Then
          kssas_SerialNumberOffsetFromSEofNW = Null    '  for most VBasics  CVErr(xlErrNA)  for
Excel
        Else
          IntXY.X = varTemp7
          IntXY.Y = varTemp8
          LSet LngNum = IntXY
          kssas_SerialNumberOffsetFromSEofNW = LngNum.a
    End If
End Function

Function kssas_SubSectABCD_FromColumnRow(ByVal Column As Single, ByVal Row As Single) As Varian
t
'' Returns a USGS/BLM ACDB style subsection form a kssas column,row pair.
'' Greg Pouch 95 03 10  KGS

Dim String1 As String, String2 As String, strHold As String
Dim j As Long, length As Long
    String1 = kssas_SubSectNWofNEFromColumnRow(Column, Row)
    length = Len(String1)
    String2 = ""
    For j = 1 To length Step 2
```

```
        strHold = Mid$(String1, j, 2)
        Select Case strHold
            Case "NE"
                String2 = "A" & String2
            Case "NW"
                String2 = "B" & String2
            Case "SW"
                String2 = "C" & String2
            Case "SE"
                String2 = "D" & String2
            Case Else
                String2 = "x" & String2
        End Select
    Next j
    kssas_SubSectABCD_FromColumnRow = String2

End Function

Function kssas_SubSectABCDFromSerialNumber(ByVal SerialNumber As Long) As Variant
'' Returns a USGS/BLM DCBA style subsection form a kssas serial number.
'' Greg Pouch 95 03 10  KGS


Dim IntXY As typIntXY, LngNum As typLong, ColNum As Integer, RowNum As Integer
    LngNum.a = SerialNumber
    LSet IntXY = LngNum
    ColNum = IntXY.X
    RowNum = IntXY.Y
    kssas_SubSectABCDFromSerialNumber = kssas_SubSectABCD_FromColumnRow(ColNum, RowNum)
End Function

Function kssas_SubSectNWofNEFromColumnRow(ByVal Column As Single, ByVal Row As Single) As Varia
nt
'' Returns a NW1/4 of SE1/4 style subsection form a kssas column,row pair.
'' Greg Pouch 95 03 10  KGS
'Note that this assumes that it can get the subsection offsets by simple division
'It will also only do a small number of quarterings, specifically, Log_2(SECTION_STEP)-1
Dim RowOffset As Single, ColOffset As Single
'RowOffset and ColOffset should be between 0 and SECTION_STEP
    RowOffset = Row - Int(Row / SECTION_STEP) * SECTION_STEP
    ColOffset = Column - Int(Column / SECTION_STEP) * SECTION_STEP

Dim strNS As String, strEW As String, strOut As String
Dim Threshold As Integer, intTemp1 As Integer, intTemp2 As Integer, n As Integer
Threshold = SECTION_STEP / 2
 strEW = ""
 strNS = ""
 strOut = ""
Do
' Debug.Print ColOffset, Threshold, strEW
 If ColOffset = Threshold Then Exit Do
 If ColOffset > Threshold Then
     strEW = strEW & "E"
     ColOffset = ColOffset - Threshold
   Else
     strEW = strEW & "W"
 End If
 Threshold = Threshold / 2
' Debug.Print ColOffset, Threshold, strEW
' Debug.Print
Loop While Threshold > 1


Threshold = SECTION_STEP / 2
Do
 If RowOffset = Threshold Then Exit Do
 If RowOffset > Threshold Then
   strNS = strNS & "N"
   RowOffset = RowOffset - Threshold
```

```
  Else
     strNS = strNS & "S"
 End If
 Threshold = Threshold / 2
Loop While Threshold > 1
'This will definitely work for quarterings, it might work for halvings.  GWP 950310
Dim strTemp9 As String
strOut = ""
intTemp1 = Len(strNS)
intTemp2 = Len(strEW)
If intTemp1 < intTemp2 Then intTemp1 = intTemp2
For n = 0 To intTemp1 - 1
  strTemp9 = Mid$(strNS, intTemp1 - n, 1)
  If strTemp9 = "" Then
     strOut = strOut & " "
    Else
     strOut = strOut & strTemp9
  End If
  strTemp9 = Mid$(strEW, intTemp1 - n, 1)
    If strTemp9 = "" Then
     strOut = strOut & " "
    Else
     strOut = strOut & strTemp9
  End If
' strOut = strOut & Mid$(strNS, intTemp1 - n, 1) & Mid$(strEW, intTemp1 - n, 1)
Next n

kssas_SubSectNWofNEFromColumnRow = strOut

End Function

Function kssas_SubSectNWofNEFromSerialNumber(ByVal SerialNumber As Long) As Variant
'' Returns a NW1/4 of SE1/4 style subsection form a kssas column,row pair.
'' Greg Pouch 95 03 10  KGS

Dim IntXY As typIntXY, LngNum As typLong, ColNum As Integer, RowNum As Integer
    LngNum.a = SerialNumber
    LSet IntXY = LngNum
    ColNum = IntXY.X
    RowNum = IntXY.Y
    kssas_SubSectNWofNEFromSerialNumber = kssas_SubSectNWofNEFromColumnRow(ColNum, RowNum)
End Function

'^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
'------------------------------------------------------------------------
'vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
Function kssas_TierFromRow(ByVal RowNum As Integer) As Variant
'' Returns the tier in nnD format (like 35S or 01N) from a kssas row number
'' Greg Pouch  KGS 95 03 07


Dim TierNum As Integer
If RowNum < 8832 Then
    'South of 6th Baseline
    TierNum = OFFSET_TIER - (RowNum \ SECTION_STEP - OFFSET_NORTH) \ 6
    kssas_TierFromRow = Format$(TierNum, "00") & "S"
   Else
     'North of 6th Baseline
    TierNum = 1 - (OFFSET_TIER - (RowNum \ SECTION_STEP - OFFSET_NORTH) \ 6)
    kssas_TierFromRow = Format$(TierNum, "00") & "N"
End If
           '=IF((kssas_Southing-2399.999999)/192>0,TEXT(INT((kssas_Southing-2208)/192),"00") &
 "S",TEXT((2400-kssas_Southing)/(192),"00")&"N")
End Function

'^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
'------------------------------------------------------------------------
'vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
Function kssas_TTdRRdSSFromColumnRow(ByVal ColNum As Integer, ByVal RowNum As Integer) As Varia
```

```
nt
'' Returns the complete SECTION name (like 09S13W35) from  a kssas column,row pair.
'' Greg Pouch 95 03 10  KGS
  kssas_TTdRRdSSFromColumnRow = kssas_TierFromRow(RowNum) & kssas_RangeFromColumn(ColNum) & For
mat$(kssas_SectionFromColumnRow(ColNum, RowNum), "00")
End Function

'^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
'---------------------------------------------------------------------------
'vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
Function kssas_TTdRRdSSFromSerialNumber(ByVal SerialNumber As Long) As Variant
'' Returns the complete SECTION name (like 09S13W35) from  a kssas serial number.
'' Greg Pouch 95 03 10  KGS
Dim IntXY As typIntXY, LngNum As typLong, ColNum As Integer, RowNum As Integer
    LngNum.a = SerialNumber
    LSet IntXY = LngNum
    ColNum = IntXY.X
    RowNum = IntXY.Y
    kssas_TTdRRdSSFromSerialNumber = kssas_TTdRRdSSFromColumnRow(ColNum, RowNum)
End Function

Function myATAN2(ByVal X As Single, ByVal Y As Single)
Dim Temp As Single
Const PI = 3.14159265358979
If X = 0 Then
    If Y > 0 Then
        myATAN2 = 0
      Else
        myATAN2 = PI
    End If
    Exit Function
End If

Temp = Atn(Y / X)
If X < 0 Then
     myATAN2 = PI + Temp
 Else
   If Y > 0 Then
      myATAN2 = Temp
     Else
      myATAN2 = 2 * PI + Temp
   End If
End If

End Function
```

```
Option Explicit


';Type RealPoint
';    X As Single
';    Y As Single
';End Type
'Global Const SECTION_STEP = 32
'The SECTION_STEP=32 is used to allow subdivision into smaller areas and still use short intege
rs.
' For use of the quartering options, it must be a power of two.
' If SECTION_STEP=2**N, you can have N-1 levels of halving/quartering and still have an integer
Const OFFSET_RANGE = 44, OFFSET_EAST = 600
'  OFFSET_RANGE specifies that the counting starts from Range 44W
'  OFFSET_EAST specifies the false easting (in miles) added to the baseline (at 44W)
'  Note that OFFSET_EAST  MUST be a multiple of 6 for the inverse routines to work.
Const OFFSET_TIER = 36, OFFSET_NORTH = 60
' specifies that T36S will have false northing of 60 miles
'  Note that OFFSET_NORTH  MUST be a multiple of 6 for the inverse routines to work.
'    (northings in 1/32th miles become 4 digit numbers for whole state)



'LEO -like functions to return centerpoints of legal locations.          GWP 950310
'PERSONAL DISCLAIMER.  These methods will be based on the assumption     GWP 950310
' that a section is bounded by four points, when it is really defined    GWP 950310
' by eight points                                                        GWP 950310
'   (the four corner points and the four midpoints of the outlines       GWP 950310
'    are all surveyed locations and should be used in these computations)GWP 950310
' However, KGS presently lacks the data to do this the right way, so     GWP 950310
'  we'll do it the quick way instead.                                    GWP 950310
'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx


'Profile string functions.  Profile int functions not declared, because you
'    can use the string functions, then val them.
'
Declare Function GetProfileString Lib "Kernel" (ByVal lpAppName As String, lpKeyName As Any, By
Val lpDefault As String, ByVal lpReturnedString As String, ByVal nSize As Integer) As Integer
'Declare Function GetPrivateProfileString Lib "Kernel" (ByVal lpApplicationName As String, lpKe
yName As Any, ByVal lpDefault As String, ByVal lpReturnedString As String, ByVal nSize As Integ
er, ByVal lpFileName As String) As Integer
Declare Function GetPrivateProfileString Lib "Kernel" (ByVal lpSectionName As String, ByVal lpK
eyName As String, ByVal lpDefault As String, ByVal lpReturnedString As String, ByVal nSize As I
nteger, ByVal lpFileName As String) As Integer
Declare Function WriteProfileString Lib "Kernel" (ByVal lpApplicationName As String, lpKeyName
As Any, lpString As Any) As Integer
Declare Function WritePrivateProfileString Lib "Kernel" (ByVal lpApplicationName As String, lpK
eyName As Any, lpString As Any, ByVal lplFileName As String) As Integer

'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Function ATAN2(ByVal X As Single, ByVal Y As Single)
'' Not so cheap version of the fortran ATAN2 function in VBasic
''Greg Pouch 95 03 10  KGS


Dim Temp As Single
Const PI = 3.14159265358979
If X = 0 Then
    If Y > 0 Then
        ATAN2 = 0
      Else
        ATAN2 = PI
    End If
    Exit Function
End If

Temp = Atn(Y / X)
```

```
If X < 0 Then
     ATAN2 = PI + Temp
 Else
   If Y > 0 Then
      ATAN2 = Temp
     Else
      ATAN2 = 2 * PI + Temp
   End If
End If

End Function

Sub dXdYFromXY_Outline_LOOP(XYCoords() As Single, ByVal XIn As Single, ByVal YIn As Single, dYO
ut As Single, dXOut As Single, QuarteringWanted As Integer)
''  Finds the "subdivision" corresponding to the location X Y , given the section outline
'' VERY SLOW.   It does by finding X,Y given the subdivision for each possible subdivision
'' and picking the closest one.
''
''Greg Pouch 95 03 13  KGS

'Column and Row are the KSSAS column and row numbers.  They can be complete locations or just o
ffsets.
'The two dimensional array XYPoints contains the XY coordinates of the outline
' of the section.  They are to be stored as X1,Y1, X2,Y2, X3, Y3 , X4, Y4, ....
Dim NWCorner As RealPoint, NECorner As RealPoint
Dim SWCorner As RealPoint, SECorner As RealPoint, BadCenter As RealPoint
Dim jError As Long

Dim XOut As Single, YOut As Single, nRow As Long, nCol As Long
Dim nRowMin As Long, nColMin As Long, dX As Single, dY As Single
Dim Dist As Single, MinDist As Single, ArrayOut As Variant, SectionStep As Integer
   Call GetCornersOfSection(XYCoords(), NECorner, NWCorner, SWCorner, SECorner, BadCenter, jErr
or) '
MinDist = 1E+34
'A really brute force approach to this problem.  Divide the section
'into however many chunks SectionStep lets you cut it and still use integers,
' then check each point for the distance to the target.
SectionStep = 2 ^ (QuarteringWanted + 1)
7 For nRow = 1 To SectionStep - 1
    dY = nRow / SectionStep
    For nCol = 1 To SectionStep - 1
        dX = nCol / SectionStep
            XOut = (1 - dY) * ((1 - dX) * SWCorner.X + (dX) * SECorner.X) + (dY) * ((1 - dX) *
NWCorner.X + (dX) * NECorner.X)
            YOut = (1 - dX) * ((1 - dY) * SWCorner.Y + (dY) * NWCorner.Y) + (dX) * ((1 - dY) *
SECorner.Y + (dY) * NECorner.Y)
            Dist = (XIn - XOut) * (XIn - XOut) + (YIn - YOut) * (YIn - YOut)
            If Dist < MinDist Then
                MinDist = Dist
                nRowMin = nRow
                nColMin = nCol
            End If
    Next nCol
Next nRow

  dXOut = nColMin
  dYOut = nRowMin

End Sub

Function dXFromABCD(ByVal SubSect As String) As Variant
'' This function returns the legXY  DX column offset (fractional section east of the west edge)
''  from a BLM/USGS style ABBDC style subsection.  Add to section columns

'' Greg Pouch  KGS 95 03 20
SubSect = UCase$(Trim$(SubSect))

Dim CurCol As Single, CurPower As Integer, n As Long, n2 As Long
CurCol = 1# / 2#  'Start in the middle
```

```
For n = 1 To Len(SubSect)
    Select Case Mid$(SubSect, n, 1)
       Case "A", "D" 'in the east half, so add
          CurCol = CurCol + 1# / 2 ^ (n + 1)
       Case "B", "C" 'in west half, so subtract
          CurCol = CurCol - 1# / 2 ^ (n + 1)
       Case Else
          dXFromABCD = Null    '  for most VBasics  CVErr(xlErrNA)  for Excel
          Exit Function
    End Select
Next n
dXFromABCD = CurCol

End Function

Function dXFromSubsectSEofNW(ByVal SubSect As String) As Variant
'' Returns a legXY DX column offset from a small to big  NE1/4 of NW1/4 of .... style subsectio
n
'' Greg Pouch  KGS 95 03 20
'' Calling routine is responsible for trimming spaces, but must be
'' careful to do so properly.  EW must be in even numbered locations, NS must be in odd-nummber
ed locations.
SubSect = UCase$((SubSect))

Dim CurCol As Single, CurPower As Integer, n As Long, n2 As Long
CurCol = 1# / 2#  'Start in the middle
CurPower = Len(SubSect) / 2 + 1
For n = 1 To Len(SubSect) Step 2
    Select Case Mid$(SubSect, n + 1, 1)
       Case "E" 'in the east half, so add
          CurCol = CurCol + 1# / 2 ^ CurPower
       Case "W" 'in west half, so subtract
          CurCol = CurCol - 1# / 2 ^ CurPower
       Case " ", "_", "C" 'Ignore these characters

       Case Else
          dXFromSubsectSEofNW = Null    '  for most VBasics  CVErr(xlErrNA)  for Excel
          Exit Function
    End Select
    CurPower = CurPower - 1
Next n
dXFromSubsectSEofNW = CurCol

End Function

Function dYFromSubSectABCD(ByVal SubSect As String) As Variant
'' Returns a legXY DY Row offset from a USGS/BLM style aadcb style subsection

'' Greg Pouch  KGS 95 03 07
SubSect = UCase$(Trim$(SubSect))

Dim CurRow As Single, n As Long
CurRow = 1# / 2#  'Start in the middle
For n = 1 To Len(SubSect)
    Select Case Mid$(SubSect, n, 1)
       Case "A", "B" 'in the north half, so add
          CurRow = CurRow + 1# / 2 ^ (n + 1)
       Case "C", "D" 'in soth half, so subtract
          CurRow = CurRow - 1# / 2 ^ (n + 1)
       Case Else
          dYFromSubSectABCD = Null    '  for most VBasics  CVErr(xlErrNA)  for Excel ' Null  '
for most VBasics  CVErr(xlErrNA)  for Excel
          Exit Function
    End Select
Next n
dYFromSubSectABCD = CurRow

End Function
```

```
Function dYFromSubsectSEofNW(ByVal SubSect As String) As Variant
'' Returns a legXY DY row offset from a small to big  NE1/4 of NW1/4 of .... style subsection
'' Greg Pouch  KGS 95 03 07

SubSect = UCase$(Trim$(SubSect))

Dim CurRow As Single, CurPower As Integer, n As Long, n2 As Long
CurRow = 1# / 2#  'Start in the middle
CurPower = Len(SubSect) / 2 + 1
For n = 1 To Len(SubSect) Step 2
    Select Case Mid$(SubSect, n, 1)
       Case "N" 'in the north half, so add
          CurRow = CurRow + 1# / 2 ^ CurPower
       Case "S" 'in south half, so subtract
          CurRow = CurRow - 1# / 2 ^ CurPower
       Case " ", "_", "C"
       Case Else
          dYFromSubsectSEofNW = Null    '  for most VBasics  CVErr(xlErrNA)  for Excel
          Exit Function
    End Select
    CurPower = CurPower - 1
Next n
dYFromSubsectSEofNW = CurRow

End Function

Function FixSubSectionName(ByVal InName As String) As String
    InName = UCase$(InName)
    'Make sure we have an even number of characters, and replace blanks with underscores.
    If InName <> "" Then
        If Len(InName) Mod 2 Then InName = InName & " "
            Do While InStr(InName, " ")
            Mid$(InName, InStr(InName, " "), 1) = "_"
            Loop
    End If
    FixSubSectionName = InName
End Function

'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Sub GetBoundBox(XYCoords() As Single, MinX As Single, MinY As Single, MaxX As Single, MaxY As S
ingle)
'' Finds the bounding box (MinX,MinY...) around a set of real*4 coordinates stored as an XY Arr
ays
''Greg Pouch 95 03 10  KGS
Dim nFirstPoint As Long, nLastPoint As Long, nGoodPoints As Long
Dim nPt As Long, nPoints As Long, X_INDEX As Long, Y_INDEX As Long
Y_INDEX = UBound(XYCoords, 2):  X_INDEX = LBound(XYCoords, 2)
  MinX = 1E+34: MaxX = -1E+34
  MinY = 1E+34: MaxY = -1E+34


   For nPt = LBound(XYCoords, 1) To UBound(XYCoords, 1)
     If (XYCoords(nPt, X_INDEX)) <> 0 Then
        'This will hopefully AVOID problems caused by uninitialized arrays.
        'This will CAUSE a problem if the user's coordinate system includes the origin.
          If XYCoords(nPt, X_INDEX) < MinX Then MinX = XYCoords(nPt, X_INDEX)
          If XYCoords(nPt, Y_INDEX) < MinY Then MinY = XYCoords(nPt, Y_INDEX)
          If XYCoords(nPt, X_INDEX) > MaxX Then MaxX = XYCoords(nPt, X_INDEX)
          If XYCoords(nPt, Y_INDEX) > MaxY Then MaxY = XYCoords(nPt, Y_INDEX)
      End If
  Next nPt
End Sub

'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Sub GetCornersOfSection(XYCoords() As Single, NECorner As RealPoint, NWCorner As RealPoint, SWC
orner As RealPoint, SECorner As RealPoint, BadCenter As RealPoint, jErr As Long)
'' Pulls the NW,SW,NE,SE corners from an outline stored as a two dimensional array
'' of singles in the form X(nPt)=XY(nPt,1),Y(nPt)=XY(nPt,2)
```

```
'' It does this by finding the center point as the average in X and the average in Y
'' of the bounding box, calculates the (math) bearing to each point,
'' then finds the angle closest to pi/4, 3pi/4, 5pi/4, and 7pi/4

''Greg Pouch 95 03 10  KGS
  Dim nFirstPoint As Long, nLastPoint As Long, nGoodPoints As Long
  Dim nPt As Long, nPoints As Long, X_INDEX As Long, Y_INDEX As Long

  jErr = 0
  Y_INDEX = UBound(XYCoords, 2):  X_INDEX = LBound(XYCoords, 2)
  If Y_INDEX - X_INDEX <> 1 Then
 '    'Too many or too few subscripts
     jErr = 2
     Exit Sub
  End If

  nLastPoint = UBound(XYCoords, 1):  nFirstPoint = LBound(XYCoords, 1)
  nPoints = nLastPoint - nFirstPoint + 1
  If nPoints < 4 Then
     'Too few points
     jErr = 4
     Exit Sub
  End If



  Dim MinX As Single, MaxX As Single, MinY As Single, MaxY As Single
  Call GetBoundBox(XYCoords(), MinX, MinY, MaxX, MaxY)
  BadCenter.X = (MinX + MaxX) / 2
  BadCenter.Y = (MinY + MaxY) / 2
  'At this point, BadCenter contains the average of the extrema
  '  This may be close to the center, or it may not.  Improved accuracy and
  '  generality could be had by using some centroid calculation.

'Now, find the angular location of each point relative to the "center"
  ReDim Angles(nFirstPoint To nLastPoint) As Single
  Dim Theta As Single
  'Rather than doing these trig calculations four times, and searching each time,
  '  do the trig once and search four times.
  For nPt = nFirstPoint To nLastPoint
    If (XYCoords(nPt, X_INDEX)) <> 0 Then
        Angles(nPt) = ATAN2((XYCoords(nPt, X_INDEX) - BadCenter.X), (XYCoords(nPt, Y_INDEX) - B
adCenter.Y))
      Else
        Angles(nPt) = 1E+32  'This keeps zero data from confusing IndexOfElementClosestTo
    End If
  Next nPt
Dim PI_OVER_4 As Single
PI_OVER_4 = 3.14159265358979 / 4

  ' Northeast corner is at pi/4
  nPt = IndexOfElementClosestTo(Angles(), PI_OVER_4)
  NECorner.X = XYCoords(nPt, X_INDEX): NECorner.Y = XYCoords(nPt, Y_INDEX)

  ' Northwest corner is at 3pi/4
  nPt = IndexOfElementClosestTo(Angles(), 3 * PI_OVER_4)
  NWCorner.X = XYCoords(nPt, X_INDEX): NWCorner.Y = XYCoords(nPt, Y_INDEX)

  ' Southwest corner is at 5pi/4
  nPt = IndexOfElementClosestTo(Angles(), 5 * PI_OVER_4)
  SWCorner.X = XYCoords(nPt, X_INDEX): SWCorner.Y = XYCoords(nPt, Y_INDEX)

  ' Southeast corner is at 7pi/4
  nPt = IndexOfElementClosestTo(Angles(), 7 * PI_OVER_4)
  SECorner.X = XYCoords(nPt, X_INDEX): SECorner.Y = XYCoords(nPt, Y_INDEX)

End Sub

'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
Function IndexOfElementClosestTo(sngData() As Single, Target As Single)
'' This function gives back the array index of the element in sngData
'' that is closest to the Target value
''Greg Pouch 95 03 10  KGS
  Dim nClosest As Long, Temp As Single, n As Long, MinSoFar
  MinSoFar = 1E+34
  For n = LBound(sngData) To UBound(sngData)
    Temp = Abs(sngData(n) - Target)
    If Temp < MinSoFar Then
        MinSoFar = Temp
        nClosest = n
    End If
  Next n
  IndexOfElementClosestTo = nClosest

End Function

Sub OffsetCornrFromXY_Outln(XYCoords() As Single, ByVal XIn As Single, ByVal YIn As Single, Eas
ting As Single, Northing As Single, Corner As String)
''A subroutine to convert XY Coordinates to offsets from a section corner to "footages"
''These are traditionally called footages, but this routine will not do the
'' unit conversion for you.   If you are using meter coordinates for XYCoords(),
'' then Easting and Northing will be in METERS.  If XYCoords are in Feet, then Easting and Nort
hing will be in FEET.
''THE CALLING ROUTINE/PROGRAM MUST TAKE CARE OF UNIT CONVERSIONS.
''
''Inputs are the XY Coordinates of the outline of the section
'' and the XY Location of the target point.  If you wish to use
'' a particular corner, enter that in Corner.  If corner is left blank,
''  this routine will pick the nearest corner and use that for the references.
''The corner footages are referenced to ( NW, SW, NE, or NW)
''The distances in units constistent with XYCoords along the section edges.
''The offsets will be made parallel to section edges.
''Greg Pouch 95 03 14  KGS

Dim NW As RealPoint, NE As RealPoint
Dim SW As RealPoint, SE As RealPoint, BadCenter As RealPoint
Dim dE As RealPoint, dN As RealPoint, StartAt As RealPoint
'dE and dN will be unit vectors pointing in the directions to take the offsets.
Dim jError As Long
   Call GetCornersOfSection(XYCoords(), NE, NW, SW, SE, BadCenter, jError) '

   Dim D As Single, MinDist As Single
   MinDist = 1E+34
   Corner = UCase$(Trim$(Corner))
If Corner = "" Then
'Only choose corner if blank.  Otherwise, use the corner specified by user.

  'Initialize to NW Corner
  MinDist = (XIn - NW.X) * (XIn - NW.X) + (YIn - NW.Y) * (YIn - NW.Y): Corner = "NW"
  'Look for closest corner
  D = (XIn - NE.X) * (XIn - NE.X) + (YIn - NE.Y) * (YIn - NE.Y)
  If D < MinDist Then MinDist = D: Corner = "NE"

  D = (XIn - SE.X) * (XIn - SE.X) + (YIn - SE.Y) * (YIn - SE.Y)
  If D < MinDist Then MinDist = D: Corner = "SE"

  D = (XIn - SW.X) * (XIn - SW.X) + (YIn - SW.Y) * (YIn - SW.Y)
  If D < MinDist Then MinDist = D: Corner = "SW"

End If

Select Case Corner
     Case "SW" 'Use SW corner for reference, so the south and west edges will be dE and dN resp
ectively
       StartAt.X = SW.X: StartAt.Y = SW.Y
       dE.X = SE.X - SW.X: dE.Y = SE.X - SW.Y
       dN.X = NW.X - SW.X: dN.Y = NW.Y - SW.Y
     Case "SE" 'Use SE, NE, and SW Corners
```

```
       StartAt.X = SE.X: StartAt.Y = SE.Y
       dE.X = SW.X - SE.X: dE.Y = SW.Y - SE.Y
       dN.X = NE.X - SE.X: dN.Y = NE.Y - SE.Y
     Case "NW"  'Use NW,SW, and NE Corners
       StartAt.X = NW.X: StartAt.Y = NW.Y
       dE.X = NE.X - NW.X: dE.Y = NE.Y - NW.Y
       dN.X = SW.X - NW.X: dN.Y = SW.Y - NW.Y
     Case "NE" 'Use NE, NW, and SE Corners
       StartAt.X = NE.X: StartAt.Y = NE.Y
       dE.X = NW.X - NE.X: dE.Y = NW.Y - NE.Y
       dN.X = SE.X - NE.X: dN.Y = SE.Y - NE.Y
End Select
'Normalize by length to make unit vectors
   D = Sqr(dN.X * dN.X + dN.Y * dN.Y)
   dN.X = dN.X / D: dN.Y = dN.Y / D
   D = Sqr(dE.X * dE.X + dE.Y * dE.Y)
   dE.X = dE.X / D: dE.Y = dE.Y / D
   'Solve by Cramer's rule   yech
   XIn = XIn - StartAt.X
   YIn = YIn - StartAt.Y
   Dim detA As Single, N1 As Single, n2 As Single
   detA = dE.X * dN.Y - dE.Y * dN.X
   N1 = XIn * dN.Y - YIn * dN.X
   n2 = YIn * dE.X - XIn * dE.Y
   Easting = N1 / detA
   Northing = n2 / detA

End Sub

Function XYFrom_OutlineCornrOffst(XYCoords() As Single, ByVal Corner As String, ByVal Easting A
s Single, ByVal Northing As Single, XOut As Single, YOut As Single) As Long
''A subroutine to convert offsets from a section corner to XY Coordinates
''These are traditionally called footages, but this routine will not do the
'' unit conversion for you.   If you are using meter coordinates for XYCoords(),
'' then Easting and Northing must be in METERS.  If XYCoords are in Feet, then Easting and Nort
hing must be in FEET.
''THE CALLING ROUTINE/PROGRAM MUST TAKE CARE OF UNIT CONVERSIONS.
''
''Inputs are the XY Coordinates of the outline of the section
''The corner footages are referenced to ( NW, SW, NE, or NW)
''The distances in units consistent with XYCoords along the section edges.
''The offsets will be made parallel to section edges.


'GWPouch 960402 Changed from Sub to Function.  true indicates success, False indiciates failure
.


''Greg Pouch 95 03 14  KGS

Dim NW As RealPoint, NE As RealPoint
Dim SW As RealPoint, SE As RealPoint, BadCenter As RealPoint
Dim dE As RealPoint, dN As RealPoint, StartAt As RealPoint
'dE and dN will be unit vectors pointing in the directions to take the offsets.
Dim jError As Long
   Call GetCornersOfSection(XYCoords(), NE, NW, SW, SE, BadCenter, jError) '
      If jError <> 0 Then
         XYFrom_OutlineCornrOffst = False
         Exit Function
      End If

   Corner = UCase$(Trim$(Corner))
   Select Case Corner
     Case "SW" 'Use SW corner for reference, so the south and west edges will be dE and dN resp
ectively
       StartAt.X = SW.X: StartAt.Y = SW.Y
       dE.X = SE.X - SW.X: dE.Y = SE.Y - SW.Y
       dN.X = NW.X - SW.X: dN.Y = NW.Y - SW.Y
     Case "SE" 'Use SE, NE, and SW Corners
```

```
       StartAt.X = SE.X: StartAt.Y = SE.Y
       dE.X = SW.X - SE.X: dE.Y = SW.Y - SE.Y
       dN.X = NE.X - SE.X: dN.Y = NE.Y - SE.Y
     Case "NW"  'Use NW,SW, and NE Corners
       StartAt.X = NW.X: StartAt.Y = NW.Y
       dE.X = NE.X - NW.X: dE.Y = NE.Y - NW.Y
       dN.X = SW.X - NW.X: dN.Y = SW.Y - NW.Y
     Case "NE" 'Use NE, NW, and SE Corners
       StartAt.X = NE.X: StartAt.Y = NE.Y
       dE.X = NW.X - NE.X: dE.Y = NW.Y - NE.Y
       dN.X = SE.X - NE.X: dN.Y = SE.Y - NE.Y
     Case Else
         XYFrom_OutlineCornrOffst = False
         Exit Function
   End Select
   Dim D As Single
'Normalize by length to make unit vectors
   D = Sqr(dN.X * dN.X + dN.Y * dN.Y)
   dN.X = dN.X / D: dN.Y = dN.Y / D
   D = Sqr(dE.X * dE.X + dE.Y * dE.Y)
   dE.X = dE.X / D: dE.Y = dE.Y / D

    XOut = StartAt.X + Easting * dE.X + Northing * dN.X
    YOut = StartAt.Y + Easting * dE.Y + Northing * dN.Y

    XYFrom_OutlineCornrOffst = True

End Function

Function XYFromDXDYQuad(ByVal dX As Single, dY As Single, XYCoords() As Single, XOut As Single,
 YOut As Single) As Long
''  Converts from dX , dY to X,Y  , given the section outline.
'' dX is the fractional section east of west line and should be in range 0-1
'' dY is fractional section north of south line and should alse be in rage 0-1
''Greg Pouch 95 03 20 KGS
''
''
'' dX and dY are the legXY dX and dY offsets in X and Y ,respectively.
'The two dimensional array XYPoints contains the XY coordinates of the outline
' of the section.  They are to be stored as X1,Y1, X2,Y2, X3, Y3 , X4, Y4, ....
Dim NWCorner As RealPoint, NECorner As RealPoint
Dim SWCorner As RealPoint, SECorner As RealPoint, BadCenter As RealPoint
Dim jError As Long
   'if dX<0 or dY<0 or dX>1 or dY> 1 then jErr=-1
   Call GetCornersOfSection(XYCoords(), NECorner, NWCorner, SWCorner, SECorner, BadCenter, jErr
or) '
      If jError <> 0 Then
         XYFromDXDYQuad = False
         Exit Function
      End If



   XOut = (1 - dY) * ((1 - dX) * SWCorner.X + (dX) * SECorner.X) + (dY) * ((1 - dX) * NWCorner.
X + (dX) * NECorner.X)
   YOut = (1 - dX) * ((1 - dY) * SWCorner.Y + (dY) * NWCorner.Y) + (dX) * ((1 - dY) * SECorner.
Y + (dY) * NECorner.Y)
   XYFromDXDYQuad = True
End Function
```

```
Option Explicit
''A set of subroutines and functions for handling XYBLOBS
'' An XY BLOB is a string of real*4 X,Y Pairs stored
'' as a Binary Large OBject in a database.
'' If you could EQUIVALENCE in Basic, you would
'' not need this module, you would just equivalence the
'' XYBLOB to an array of RealPoints and be done with it.
'' (A REALPOINT is a pair of real*4s aka singles aka floats
''   storing an XY pair for a point.)
'
'' Greg Pouch  Kansas Geological Survey
'' Last modified 95 03 23

Type RealPoint
    X As Single
    Y As Single
End Type
Type String8
    f As String * 8
End Type
Dim RlPt As RealPoint
Dim S8 As String8
'File Open/Save Dialog Flags
Global Const OFN_READONLY = &H1&
Global Const OFN_OVERWRITEPROMPT = &H2&
Global Const OFN_HIDEREADONLY = &H4&
Global Const OFN_NOCHANGEDIR = &H8&
Global Const OFN_SHOWHELP = &H10&
Global Const OFN_NOVALIDATE = &H100&
Global Const OFN_ALLOWMULTISELECT = &H200&
Global Const OFN_EXTENSIONDIFFERENT = &H400&
Global Const OFN_PATHMUSTEXIST = &H800&
Global Const OFN_FILEMUSTEXIST = &H1000&
Global Const OFN_CREATEPROMPT = &H2000&
Global Const OFN_SHAREAWARE = &H4000&
Global Const OFN_NOREADONLYRETURN = &H8000&

' Field Data Types
Global Const DB_BOOLEAN = 1
Global Const DB_BYTE = 2
Global Const DB_INTEGER = 3
Global Const DB_LONG = 4
Global Const DB_CURRENCY = 5
Global Const DB_SINGLE = 6
Global Const DB_DOUBLE = 7
Global Const DB_DATE = 8
Global Const DB_TEXT = 10
Global Const DB_LONGBINARY = 11
Global Const DB_MEMO = 12

Sub BLOB2RealPoints(BLOBin As String, DataPts() As RealPoint)
Dim n As Long, nPoints As Long, NPos As Long
nPoints = Len(BLOBin) / 8 '8 Bytes to a RealPoint
Dim NStart As Long
'We will append datapoints to the end of the current array.
' This way, we can have an unlimited number (kind of) of points
' in a line.  Before starting to fill this array using
' this subroutine, you should redim it to contain zero points
If LBound(DataPts) <> UBound(DataPts) Then
    NStart = UBound(DataPts) + 1
    ReDim Preserve DataPts(1 To UBound(DataPts) + nPoints)
  Else
    NStart = 1
    ReDim Preserve DataPts(1 To nPoints)
End If

'ReDim Preserve DataPts(1 To UBound(DataPts) + NPoints)
NPos = 1  'NPos will keep track of where in the BLOB we are
For n = NStart To UBound(DataPts)
```

```
    S8.f = Mid$(BLOBin, NPos, 8)
    LSet RlPt = S8
    DataPts(n) = RlPt
    NPos = NPos + 8
Next n
End Sub

Sub BLOB2XYPairsInSnglArray(BLOBin As String, sngXYData() As Single)
'' Take XY Chains stored in a WHEAT format XYBLOB and put them into
'' a one-based, two-dimensional array.
'' sngXYData() must be a dimensionable array
''  Point nPt will be X=sngXYData(nPt,1), Y=sngXYData(nPt,1)


Dim n As Long, nPoints As Long, NPos As Long
nPoints = Len(BLOBin) / 8 '8 Bytes to a RealPoint

ReDim sngXYData(1 To nPoints, 1 To 2)
NPos = 1  'NPos will keep track of where in the BLOB we are
For n = 1 To nPoints
    S8.f = Mid$(BLOBin, NPos, 8)
    LSet RlPt = S8
    sngXYData(n, 1) = RlPt.X
    sngXYData(n, 2) = RlPt.Y
    NPos = NPos + 8
Next n

End Sub

Function CountOccurrences(ByVal MamaString As String, ByVal BabyString As String) As Long
Dim n As Long
Dim BabyLength As Long
Dim NextBabyAt As Long
    BabyLength = Len(BabyString)
    If BabyLength < 1 Then Exit Function

    Dim CurrentPos As Long
    CurrentPos = 1
    n = 0
    NextBabyAt = InStr(CurrentPos, MamaString, BabyString, 1)
    Do Until NextBabyAt = 0
         n = n + 1
         CurrentPos = NextBabyAt + BabyLength
         NextBabyAt = InStr(CurrentPos, MamaString, BabyString, 1)
    Loop
    CountOccurrences = n
End Function

Sub DebugPrintRealPoint(ThisPoint As RealPoint, strComment As String)
    Debug.Print strComment, ThisPoint.X, ThisPoint.Y
End Sub

Function MakeXYBLOBFromText(XYMemoIn As String, XYDelim As String, PairDelim As String) As Stri
ng
Dim nPoints As Long, nP As Long, n As Long, lenPairDelim As Long
Dim NextStart  As Long, NextStop As Long
On Error GoTo ERRMakeXYBLOBFromText
NextStart = 1
lenPairDelim = Len(PairDelim)

Dim Temp As String
nPoints = CountOccurrences(XYMemoIn, PairDelim)
If XYDelim = PairDelim Then nPoints = nPoints \ 2
Temp = XYMemoIn

ReDim DataPt(1 To nPoints) As RealPoint

If nPoints <= 0 Then Exit Function
For n = 1 To nPoints
```

```
    NextStop = InStr(NextStart, Temp, XYDelim)
    DataPt(n).X = Val(Mid$(Temp, NextStart, NextStop - NextStart))
    NextStart = NextStop + 1
    NextStop = InStr(NextStart, Temp, PairDelim)
    DataPt(n).Y = Val(Mid$(Temp, NextStart, NextStop - NextStart))
    NextStart = NextStop + lenPairDelim '+ 1
 'Debug.Print DataPt(n).X, DataPt(n).Y


Next n

MakeXYBLOBFromText = RealPoints2BLOB(DataPt())

Exit Function
ERRMakeXYBLOBFromText:
Select Case Err
 Case 365  'Unable to unload in this context
    Err = 0
    Exit Function
    Resume Next
 Case Else
    MsgBox "Error number " & Err & " occurred in MakeXYBlobFromText." & Chr$(13) & Chr$(10) & E
rror
    Err = 0
    Exit Function
    Resume
End Select

End Function

Function MakeXYTextFromBLOB(XYBLOBIn As String, XYDelim As String, PairDelim As String) As Stri
ng
    Dim nP As Long, nPoints As Long, XYMemoOut  As String
    nPoints = Len(XYBLOBIn) \ 8
    ReDim XYCoords(1 To 1) As RealPoint
    BLOB2RealPoints XYBLOBIn, XYCoords()
    For nP = 1 To nPoints
        XYMemoOut = XYMemoOut & XYCoords(nP).X & XYDelim & XYCoords(nP).Y & PairDelim
    Next nP
   MakeXYTextFromBLOB = XYMemoOut
End Function

Private Function MaxR4(a As Single, B As Single) As Single
    If a > B Then
        MaxR4 = a
       Else
        MaxR4 = B
    End If
End Function

Private Function MinR4(a As Single, B As Single) As Single
    If a < B Then
        MinR4 = a
       Else
        MinR4 = B
    End If
End Function

Function RealPoints2BLOB(DataPts() As RealPoint) As String
Dim n As Long
Dim BigTemp  As String
BigTemp = ""
For n = LBound(DataPts) To UBound(DataPts)
    RlPt = DataPts(n)
    LSet S8 = RlPt
    BigTemp = BigTemp & S8.f
Next n
RealPoints2BLOB = BigTemp
End Function
```

```
Function XYPairsInSnglArray2BLOB(sngXYData() As Single) As String
'' Take XY Chains stored in a two-dimensional real*4 array and put them
'' into aWHEAT format XYBLOB
''  Point nPt will be X=sngXYData(nPt,X_INDEX), Y=sngXYData(nPt,Y_INDEX)
Dim X_INDEX As Long, Y_INDEX As Long, nPt As Long
Dim nFirst As Long, nLast As Long
nFirst = LBound(sngXYData, 1): nLast = UBound(sngXYData, 1)
X_INDEX = LBound(sngXYData, 2): Y_INDEX = UBound(sngXYData, 2)
If Y_INDEX - X_INDEX > 1 Then Exit Function
Dim BigTemp  As String
BigTemp = ""
For nPt = nFirst To nLast
    RlPt.X = sngXYData(nPt, X_INDEX): RlPt.Y = sngXYData(nPt, Y_INDEX)
    LSet S8 = RlPt
    BigTemp = BigTemp & S8.f
Next nPt
XYPairsInSnglArray2BLOB = BigTemp

End Function
```