L E O I I

AUTOMATED CONVERSION

BETWEEN LEGAL AND GEOGRAPHIC REFERENCE

FOR LOCATIONS IN KANSAS

by

CHARLES G. ROSS

KANSAS GEOLOGICAL SURVEY

OPEN FILE REPORT 93 - 53

1 JUNE 1994

ABSTRACT

LEO II[1] is a second-generation software system developed for use on the PC, which is designed to convert location references accurately between legal descriptions and geographic coordinates for locations in Kansas.[2] In the LEO II system, a FORTRAN[3] program accepts locations entered directly by the user or from a selected disk file, converts from legal to geographic reference or vice-versa, and writes the converted locations to the console screen or to a second disk file. The conversion program comprises one routine that converts between reference systems, two routines that support special options, and a control routine. Potential users should be familiar with legal and geographic reference and with the PC environment in which LEO II operates. The principal functions of LEO II are available to anyone who can create an input data file in the correct form and with the proper content, with locations specified according to the standard or optional format and precision. Users with programming expertise can modify or rewrite parts of the conversion program to customize file formats, operational procedures, and program functions, and to use the alternate form for defining geographic coordinates. Reference conversion is founded on a large, "spatial" database containing the geographic locations of all Kansas section corners, which were taken from the digitized and edited section corners in the Kansas Cartographic Database (KCD). The system includes a "query" program, which may be used to examine or modify (rarely done) the contents of the LEO II database. A small set of Kansas sections is said to be "nonstandard," because of unusual size or shape. In those sections, the usual processes for subdividing sections and applying footages (methods for specifying more precise locations within a section) are not well defined. The LEO II system cannot convert locations in nonstandard sections. Attempting to convert a location in a nonstandard section is one example of errors that cause LEO II to stop performing a conversion. When error conditions end a conversion prematurely, LEO II identifies the specific error by writing an error number[4] to the terminal or output file. This documentation describes, in three chapters, the system and how to use it. The first chapter introduces concepts, functions, components, operations, and uses of the system. It is required for those users who are unfamiliar with reference systems and conversion. Chapter II reviews program-user interaction (prompts, questions, selections, commands) and details the input and output formats for batch (file-to-file) operations. The third chapter, for programmers, describes the extended options and capabilities that are accessible only by modifying the LEO II software.

---

[1]See the "Introduction" section for a derivation of the name LEO II.

[2]Previous work on converting legal coordinates in Kansas can be found in Good (1964) and later, in Morgan and McNellis (1969). Both efforts resulted in computer programs of some value.

[3] Names of programs, routines, files, databases, languages, and systems often are written in all CAPS by computer scientists, publishers, and educators. (FORTRAN 77 is used in LEO II).

[4]These are defined in Appendix A.

ACKNOWLEDGMENTS

I wish to thank the members of the Automated Cartography group, present and past, student and staff alike, for their previously unheralded contributions to the successful creation of the Kansas Cartographic Database (KCD). Without complete and accurate set of digitized corner locations for sections in Kansas, the LEO database and the LEO project would not have been possible.

David Collins spent many hours assisting in the preparation of the early drafts of this document. He performed the essential tasks of an editor: reading, analyzing, and correcting, but his efforts went beyond checks for style, grammar, and content. His extensive research and understanding of the PLS system, reference conversion, and the use of past LEO software made his comments and corrections invaluable. His research into the PLS in Kansas improved our understanding of the legal system, which, in turn, brought changes and improvements to the LEO II system.

J. C. Wong also contributed to our understanding of the realities of the PLS. His programming and application of interactive programs corrected many of the known errors in the section corner locations in the LEO II database. His achievements contributed to improvements in LEO II.

One under appreciated source of assistance in the preparation of this manuscript is the collective group of LEO II users who brought problems and deficiencies in the software to my attention in recent years. Without such feedback, the corrections might await the arrival of a new version.

I wish to thank Tom Mettille and Alan Feltz for reading the manuscript and for providing useful suggestions for changes to clarify the text from the perspective of a potential operator of LEO II (Tom), and the concerns of a programmer using or modifying the programs (Alan). They read many pages of dry and technical material, and both contributed to the final version.

Along this same vein, but in greater detail, Dan Merriam was kind enough to read the (almost) final draft, not as a user or programmer, but as an editor of professional papers. That this is a task which Dan has performed often in the past, is evident from the thoroughness, clarity, and correctness (I checked several of his marks) of his work. The text of the LEO II documentation was markedly improved by Dan's contribution, and I am grateful.

June 1994
C. Ross

TABLE OF CONTENTS

LEO II : AUTOMATED REFERENCE CONVERSION IN KANSAS

I. INTRODUCTION

The constellation of Leo comprises a specific set of stars, which are said to correspond to the figure of a lion in the sky, except in late June, when it is swallowed by the sun. The lines that define this king of the beasts are imaginary segments that connect the stars of the constellation. Assigning the lion's image to a set of points in the sky is defining a correspondence, or mapping between the lines of the lion's image and the set of points, the stars, whose locations in space coordinates are well-defined. Using the coordinates of the stars, one might be able to construct an image of the Leo lion through computer drafting, simply by entering the point coordinates of the stars (transformed to fit on a sheet of paper) and supplying connect-the-dot instructions and move (without drawing) instructions to the comnputer. And, if someone wanted to know the location of the left nostril, the existing coordinates could help to determine that location.

The LEO II software system provides a similar mapping, between the artificial shapes that are the sections and townships of Kansas - of the legal reference system - which are collections of imaginary lines that bridge specific sets of points to form (admittedly boring) quadrilaterals. As with the stars of the constellation Leo, the specific points that define the imaginary shapes[5] of sections have well-defined Earth coordinates - in the geographic reference system - commonly known as longitude and latitude. The LEO II reference conversion (software) system maps or converts reference between the legal and geographic reference systems. With this mapping, the system can create accurate maps of sections and townships, and include point or line features (in legal reference) as well. And, given a point defined (in legal reference) within the images of the sections, a fixed geographic location can be determined (and vice-versa).

This documentation describes the functions and operation of the LEO II reference conversion system, including the legal and geographic reference systems, the LEO II spatial database, the LEO II conversion program and other software, and the use of custom programming to access extended functions. The documentation is divided into three parts: (I.) Introduction to spatial reference systems, the LEO II spatial database and software, and reference system conversion; (II.) Operation of interactive and batch modes of the conversion and database query programs; and (III.) Programming information, including customized programming, extended functions and options, format control, data files, and database organization.

Users should be generally familiar with reference systems, reference conversion, and the use of the PC. Much of the necessary information is given in the introductory section. This material requires a general knowledge of the legal and geographic reference, a general understanding of

[5]As the constellations were idealized images, so were the sections of Kansas. Many people have believed (and still do believe) mistakenly that sections are squares, exactly one mile on each side. To some, sections are also laid out with the north and south sides both of constant latitude. Some even believe the east and west sides run north-to-south. These conditions are mutually exclusive.

conversion, and knowledge of some reasons for converting location references between the two. The introductory section describes reference conversion and the functions of LEO II in a non-technical way. The Operation section details using and interacting with programs for conversion and database query. The Program section informs users with minimal programming expertise how to modify programs to access new options and extended functions, otherwise unavailable.

The original LEO system (version 1) provided the first accurate conversions between the legal and geographic reference systems in late 1985 and early 1986, as described in Open File Report 89-10 (Ross, 1989b). The conversion options were restricted and the locations of section corners in the database included some erroneous coordinates, but the system produced the most accurate conversions ever done for most areas of Kansas. There were operational errors and the system could not handle footages for identifying locations within sections. When the second version of the system, LEO was released, it included new capabilities and new errors, and there were many sections that were not properly represented in the database. It made both interactive and batch operations easier and clearer; added options for footages, four corners, section side lengths, and the alternate Degrees-Minutes-Seconds (DMS) form for geographic coordinates. For LEO II, the records in the database were restructured to include additional information related to nonstandard sections. Also, the system was thoroughly documented, far beyond the original version.

Since that release, numerous problems were corrected by rewriting parts of the software, most or all of the sections with known errors were corrected in the database (Collins, 1989), most of the previously unknown "nonstandard" sections (flags) were marked in the database, and brand new documentation has been written in a simplified form. The updated software and database, with the new documentation comprise (actually, the second release of) version two of the reference system conversion package, named LEO II.

A. SYSTEM OVERVIEW

In most sets of scientific data, each element (record, sample. . .) includes at least one spatial component to identify the origin of the associated data. The spatial component is necessary for applications that depend on spatial relationships, and often serves as the "key" field, which uniquely identifies the data record or sample. For some data sets, the spatial aspect may be unimportant, non-existent, or implicit in some other aspect of the data set. For example, the name of a file may imply that all elements in the file are located in a particular place. Most geologic data, by definition or by nature, are associated fundamentally with locations on the Earth's surface. In such data sets, the most important aspect of each datum is its **location**.

The manner used to define or express this primary aspect (location) for geologic data varies, depending on the environment and the application. All of the usual methods for specifying locations involve a set of application rules and an artificial or implied mapping from a "data space" (where the data arose) to a "theoretical space," which is defined mathematically. In the context of LEO II, the data space is the surface of the Earth, while the theoretical space may take several possible forms, such as: (1) an artificial grid of longitude and latitude lines on the surface of an oblate spheroid approximating the Earth; (2) a complex map projection, like the

Modified Polyconic, defining a special Cartesian coordinate system based on cones tangent to the Earth at each latitude; or (3) an irregular grid of point locations which define the boundaries of land units (such as sections), and which constitute a crude coordinate system.

Each system of rules and the associated theoretical space defines a **system of reference**, by which locations are identified. Within a reference system, each location is specified uniquely by a set of coordinates that collectively apply only to that point. Usually, every point in the data space of a reference system can be defined (mapped to a point) in the theoretical space, but the reverse is not always true. Two reference systems with these properties are fundamental to the LEO II system. These are **geographic** reference (longitude and latitude) and **legal** reference (township, range, section,. . .). A third import system is really a limitless set of systems, it is called **projected** reference, with locations in Cartesian coordinates, as on printed maps.

Because the data space for both the geographic and legal reference systems is the surface of the Earth, it is both reasonable and possible to translate or convert locations expressed in one of the systems into the equivalent coordinates in the other system. A location specified as township, range and section can be converted into coordinates of longitude and latitude that represent the same location according to the artificial mapping between the systems. Conversely, a location in longitude and latitude can be converted into the equivalent legal location, using township, range, and section numbers, and footages from a specified corner. Conversion from geographic to legal also results in the description of a small area by successive levels of (quarter-) area subdivision (NW, SW, SE, NE) such that the point is contained in the final subdivided area. The process of changing a location, referenced by coordinates in one system, into the equivalent coordinates in another system is **reference conversion**, the principal function of LEO II.

LEO II (the name arises from the 'le' in ***le***gal and the 'eo' in g***eo***graphic) is a computer software system that can convert locations between legal reference and geographic reference. LEO II comprises a conversion program, a database of Kansas section corner locations in geographic reference (longitude, latitude), and a program for examining and modifying the database. The software is written in FORTRAN 77 for the IBM-compatible personal computer (PC), with or without a math coprocessor. With increased functionality and improved accuracy, LEO II is the second version of the LEO reference conversion system. The most recent edition of this version includes corrections of some recently discovered, minor defects in the software. The source, object, and executable files and database, fit on a 3.5-inch, high-density floppy disk.

The LEO II software comprises: (1) a program (LEO2PC) to perform the reference conversion process comprising: a large conversion routine (LEO2CVT), a control routine (LEO2PC), a routine (LEO2SETP) to set up polyconic map projection, and a routine (LEO2PRO) to project corners for finding section side lengths; (2) a database (LEO2BASE) containing the locations of all Kansas township and section corners stored in longitude and latitude by spatial relationships; and (3) a program (LEO2Q) to examine and modify the database.

There is a significant need to convert locations in Kansas from legal to geographic reference and a less significant need to convert from geographic to legal reference. In most areas of geological research, and throughout the range of state government and industrial operations there are many

historical data sets in which the data items have been located using legal reference. Many such data sets exist in the geosciences, such as the oil and gas industry, hydrologic concerns, and also in related governmental oversight and regulation functions. Because the legal reference system was land-based and required relatively simple tools, it was easier to use. And, because the legal system was the standard for property definition and other applications, nearly all historic data in Kansas were located using legal reference.

Modern applications in science, government, and industry do not use legal reference as the primary reference system for identifying locations in Kansas. Statistical analyses and graphic presentation (computer display or plotted map) of geologic data are examples of operations that can not be supported when locations are referenced in non-Cartesian systems, such as in the legal reference system. The modern use of digital computers to analyze, display, and map geologic data, which is based in large part on location and spatial relationships, is one of the many reasons for using geographic reference for data with a spatial component.

For locating a point accurately on the Earth's surface, such as for ship or airplane navigation, and many other applications, only one system of reference has been used for centuries. The most universal reference system is the "geographic" reference system, in which locations are defined by intersecting lines of longitude and latitude. Superior to the legal reference system in almost every way, geographic reference is more accurate, convenient, and exact. It has become virtually universal for locating points and lines on the Earth's surface and for defining boundaries for areas. With a computer, it is possible to convert between geographic and other reference systems, including legal (property definition) and projected (map-making) reference. Advances in computers, software, satellite positioning, and image processing further increased the use of geographic reference, a natural system for satellites. The highly accurate geographic coordinates available from the satellite-based Global Positioning System (GPS), which came on line recently, have become the de facto standard for accuracy in locating points on the Earth's surface, and to support universal, global navigation (Hurn, 1989).

A LEO II user must be familiar with geographic and legal reference, and should understand the reasons for converting between the two. The user should have a working knowledge of the PC environment: i.e., how to run programs, enter data, locate files and directories, input data by the keyboard, edit that data, identify file pathnames, print files and other basic functions. It is not necessary to have programming skills to use the basic functions of the system, but with minimal programming expertise, users may access the extended options and functions of LEO II, such as DMS (Degrees, Minutes & Seconds form for longitudes or latitudes instead of decimal degrees), four corners (to locate all four corners of the section), and (approximate) section side lengths (to calculate approximate lengths of the section sides). To use the extended options and functions, a programmer must modify or rewrite parts of the LEO II programs. Other changes can be made to the conversion program, LEO2PC, to alter the input and output formats or the process flow, thereby adapting the program to the content and form of the user's data sources and products.

The LEO II conversion process requires the spatial database, LEO2BASE, which contains all Kansas section corner locations, in geographic reference. The database was constructed with section corner locations from the Kansas Cartographic Database (KCD). The KCD comprises a

large set (2,300+) of individual, digital cartographic databases that contain map data for the state of Kansas. (The KCD includes section corners, township and range lines, hydrology, highways, county seats, railroads and other features). The KCD was constructed exclusively by use of the in-house automated cartography system, GIMMAP (jim-map, Geodata Interactive Management Map Analysis and Production), which is used to capture cartographic data, build and edit digital cartographic databases, retrieve and display maps, and support publication-quality mapping and electronic transfer of geoscience data.

The LEO2BASE database parallels the structure of the legal reference system in Kansas by containing the locations of all section corners, organized into units by township, and ordered according to spatial relationships. The section corners in Kansas are such that most sections are quadrilateral in shape, and of approximately uniform size at about one square mile. There are some areas where section boundaries do not define quads, but polygons, most commonly where surface hydrology interfered with the original survey. In other places, sections are six-sided polygons with the quarter-corners out-of-line with the corners, or even triangles. In all cases of "nonstandard" sections, the database record associated with the anomalous or irregular sections is marked to prevent LEO II from attempting reference conversion. In some future version of the system, perhaps LEO III, these unusual section boundaries and the processes for converting locations within them may be better defined.

B. REFERENCE SYSTEMS

There are two reference systems of interest to the LEO II system: the legal reference system and the geographic reference system. A third reference system, projected reference, which is actually an infinite set of parallel reference systems, was important in the construction of the LEO II database. Projected reference systems are the natural reference for maps such as the US Geological Survey (USGS) quadrangle maps, from which the locations of the section corners in Kansas were obtained for the KCD project. The locations of each section corner were digitized as projected (x,y) coordinates and converted to geographic (longitude, latitude) coordinates by PROJCT, the map projection software of GIMMAP, so they could be used to form the spatial database for LEO II.

In map production, the projected reference system is generally used. For consistent, and less distorted views of data, projected reference is also the system used for most graphic displays of cartographic and geologic data, in which quality and spatial relations are important. Statistical analyses and graphic presentation of geologic data by computer map, are examples of operations not supported in non-Cartesian systems, like legal reference (township, range, section,. . .).

The geographic reference system has long been the bedrock of worldwide navigation. Recent advances in computers, software, satellite positioning and imagery have further increased the use of this more accurate and universal reference system, in which locations are defined by pairs of coordinates that correspond to the longitude and latitude lines that intersect precisely at the point. Geographic reference offers more accuracy, convenience, and precision, and it is the universal reference system for locating points and lines, and defining areal boundaries.

Of the three reference systems used at the Kansas Geological Survey (KGS) and elsewhere in Kansas, the geographic reference system is more universal than the projected or legal reference systems, at least, for computer applications. This is because locations expressed in either of the other systems can be converted into geographic reference and back again. These operations are provided by LEO II and the PROJCT system (for map projection and deprojection). For reasons similar to those of the KGS, most users of geologic data outside the KGS rely on the geographic reference system.

1. GEOGRAPHIC REFERENCE

The geographic reference system has come to be known as the universal reference system after its many years of use around the world. Elements of the system were even used by the Greeks, prior to the Christian era (Robinson, 1984), perhaps because it is simple in design and easy to understand, visualize, and apply in a global sense.

Simplified a little, geographic reference is based on a model that includes infinitely many circles at the surface of the Earth, passing through both poles and perpendicular to the equator. These are circles of longitude. The half-circles on opposite sides of a longitude circle are "meridians" of longitude. Each meridian is assigned a unique value, which is the angular distance (as if at the center of the Earth) between the meridian and an arbitrarily specified starting meridian. Also in the geographic reference model, there is a set of circles on the Earth's surface, all parallel to the equator. Starting at the equator and progressing to the north and south poles, the "parallels" of latitude grow smaller in diameter toward the poles where they become mere points. There is no natural aspect by which circles of latitude are divided into halves as with longitudes. But, by noting the angular distance between the equator and a parallel, as it would appear to an observer at the center of the Earth, a north-south coordinate, is defined for all points along the parallel.

The equator is the circle around the largest part of the Earth. It is equally distant from the north and south poles. It divides the Earth into the northern and southern hemispheres. It is the place where the sun is directly overhead at noon on the days of the spring and autumn equinoxes. The equator is a "great circle"[6] and the primary or starting circle of latitude. All other latitudes are defined by (not great) circles parallel to the equator at the surface of the Earth. Each circle or "parallel" of latitude lies at a unique distance from the poles. Circles of latitude are parallel to each other (unlike longitude) and each is of a different size (each circle has a unique radius).

Because of the uniqueness of the equator and the poles, assigning values to circles of latitude did not require any arbitrary points of reference. The latitude at the equator was defined to be zero degrees. Above (to the north) the equator, latitudes were north latitudes; to the south they were south latitudes. The angle between the equator and either pole (as measured from the center of the Earth) is ninety degrees, so the latitude at either pole is ninety. It is usually advantageous to

[6]A great circle is formed on the surface of the Earth by any intersecting plane that passes through the center of the Earth. Great circles have the property that the shortest path between any two points on the surface of the Earth is the great circle arc between them.

have a continuous transition of latitudes from pole to pole, so, latitudes south of the equator are negative (-90 to 0), to the north, they are positive (0-90), to maintain proper spatial relationships.

Since the Earth is essentially a sphere, it was decided that there would be (as in all spheres and circles) exactly 360 degrees of longitude to circle the Earth. Circles of longitude were defined as great circles passing through both poles, and a "meridian" of longitude is a half of a circle of longitude, between the poles. But, which meridian represents which longitude? Which meridian is the first (zero degrees)? For longitudes, there was no naturally defined starating place, such as the equator was for latitudes, with the poles as natural ending points. And so, an international conference was held to determine arbitrarily, a specific and universally agreed upon, meridian at which longitudes began. The eventual compromise chose this "prime meridian" to pass through the observatory at Greenwich, England, where the longitude became zero degrees, by definition. The meridian opposite to the prime meridian lies in the vicinity of the International Date Line in the west Pacific ocean and has a longitude as 180 degrees east (or west).

Longitudes in the US are "west" longitudes. East of the prime meridian are "east" longitudes. For modern computer applications, west longitudes are given negative values, as those to the east are positive. This differentiates east and west longitudes clearly, preserves the direction of the positive x-axis for both, and properly maintains spatial relationships. If east and west longitudes were both positive, points to the east would progress from left to right when displayed/plotted, but points to the west would also progress from left to right, contrary to longitudes on the Earth. If the symbol representing Greenwich, England plotted in the center of the map, then the symbol for New York City, with a moderately large, positive longitude, would plot well to the east of Greenwich, near the Russian and Chinese border, above Pakistan.

There are two optional forms for specifying geographic coordinates for LEO II, and both are applicable for input and output. As described, the basic form for geographic reference is a pair of real values representing the longitude and latitude of the location in decimal degrees. These two values are given as "arguments" or "parameters" to the LEO II conversion routine, which expects them to be within a certain range for conversion in Kansas. Otherwise, the conversion will not be attempted. So, all latitudes must be between 36.875 and 40.125 (degrees north), and all longitudes must be between -102.125 and -94.5 (degrees west).

Expressing longitudes and latitudes as two real numbers, within the ranges specified above, is called the "decimal degrees" form for geographic reference. This is the most commonly used form for entering geographic coordinates for conversion by LEO II. In this form, the two values combine the whole degrees with a decimal fraction of degrees that represents the minutes and seconds of latitude and longitude. When legal locations are converted to geographic reference, LEO II produces two real values, the longitude and latitude, in decimal degrees.

Users with programming expertise may use another acceptable form for longitude and latitude values, each as a triplet, representing degrees, minutes, and seconds. This representation is called the "DMS" (degrees, minutes, and seconds) form or option for geographic coordinates, and is equivalent to the decimal degrees form. Converting between these optional geographic reference forms can be performed easily using a simple algorithm (see III.B.1., p42).

2. PROJECTED REFERENCE

Projected reference is an unlimited class of reference systems, which are based on the use of map projections and limitless values for parameters of the projections to define each member uniquely. Map projections transform or map the points on the surface of the Earth onto some object of special shape or some purely mathematical object space (most use cones, cylinders, or planes). Generally, the projection "object" is then assumed to be unfolded or rolled out into a planar surface, and by this, all points in geographic space are transformed or translated into a pair of cartesian coordinates (x,y), usually feet (or meters) on the ground or inches on the map.

Projected reference systems are used for the production of maps from information gathered from the real world - the surface of the Earth. Whenever a three-dimensional image is converted into a two-dimensional image through projection, something must be distorted in the projected image. The graphic images produced by map projection always have distortions in one or more aspect of direction, shape, scale, or area, but such distortions are what we are generally used to seeing. These products are how we view our world on paper. The choice of projection determines the types of distortion, allowing the mapmaker to control the kinds of distortion in maps. On this, John Wilford (1981, p.77) wrote: "The cartographer's task is to design maps that will show the least distortion or no distortion in those properties the map's intended user deems desirable."

Each projected reference system defines a two-dimensional (2D) Cartesian coordinate system, with perpendicular axes x and y. The specific nature of a projected reference system is defined by a unique map projection, an ordered translation from three-dimensional (3D) locations, such as on the Earth's surface, to a 2D surface. If the specific projection parameters are known, it is possible to "deproject" coordinates into the "universal" geographic reference system. Geographic coordinates can then be converted directly to other reference systems of interest (projected and legal reference), using LEO II and the PROJCT projection and deprojection software system at the KGS. The latter is described in KGS Open File Report 89-9 (Ross, 1989a), and its function is illustrated in part 4, Reference Conversion.

Features in the KCD, which completely covers Kansas, were digitized from the 1:24,000 scale USGS (7.5') topographic maps, because these maps were considered the highest quality source of cartographic information available. All locations in the KCD, including the section corner locations used for the LEO II database, were digitized from these USGS maps and converted accurately to digital (computer) form. Because the source maps are projections of the Earth's surface, the digitized features in the KCD are expressed in projected reference.

3. LEGAL REFERENCE

Legal reference is a complex system and methodology for locating geographic features on the Earth's surface. Identifying a point location or a contiguous area by specifying their township, range, and section (subdivisions, footages,. . .) is especially useful for describing areas in the land grid for legal or official purposes. Although many data sets using legal reference still exist, modern data collection and generation are referenced usually not to legal coordinates, but to the

geographic reference (longitude and latitude) system or a projected reference system (such as x and y in inches on a map). Locations in legal reference may be included in contemporary data sets as redundant, less accurate values. Generally, the legal reference system is no longer the preferred form for location reference in Kansas. Increasingly obsolete in the computer age, legal reference still retains some value because many historic data sets that use it still exist.

In Kansas, and elsewhere, locations have long been referenced in legal coordinates, primarily by the government. Used for many applications, "legal reference" resulted from the creation of the Public Lands Survey (PLS) by federal mandate following the Revolutionary War. The goal of the government project was to raise revenues needed to pay the huge war debts. To accomplish this, vast tracts of government (public) land were surveyed through federal funding. These tracts were parceled into officially defined properties, which were then sold to pay off the war debts. Details of the mechanisms underlying the land surveys in the Kansas Territory were provided in the 1855 Manual of Surveying Instructions (Moore, 1855), which specified procedures needed by surveyors to establish the base line, principal meridian, and correction lines for Kansas.

The U.S. Public Lands Survey System, PLSS, became the first organized, official, and systematic method for defining property (and later, adopted for defining point locations) in the country. To support the PLSS in Kansas, a land "grid" was defined and surveyed in the state, with individual units of approximately one square mile termed **sections**. Sections were surveyed in groups of thirty-six, in six rows and six columns within **townships**, whose boundaries had been surveyed. The surveys covered Kansas, moving southward from a "base line," intended to be the Nebraska border (approx. 40 degrees latitude) and "correction lines," and east and west of the "prime" (the Sixth Principal Meridian) and "guide" meridians. Surveyors located and placed physical markers on the ground for more than 80,000 section corners. This was the physical grid and the basis of a reference system, using township, range, section and subdivision, which became the standard method for identifying areas for property, taxation, and other purposes. Because it was the basis for legal property definition, the PLSS became known as the "legal reference" system.

a. Identifying The Section

Defining a location in legal reference requires a more complex set of values than in geographic reference. Unlike the fixed, pair of longitude and latitude coordinates in geographic reference, the number and type of values required to specify a legal definition is variable. In general, legal descriptions designate the township and the section within the township, in which the location to be converted exists. The township is designated by three values: the township (or row) number, the range (or column) number, and the east/west range designator (e/w indicator), identifying the range as east or west of the Sixth Principal Meridian. An additional number, from one to thirty-six, is used to specify the unique section within the chosen township.

The four values (township, range, e/w indicator, and section), used to identify the township and section, are simple and straightforward. However, identifying locations within a section, with a greater precision, involves the use of an additional set of forms and formats that complicates the specification process. Several forms of section "subdivision" plus optional selection of a "closest point" within the "target area," (the target area is what results after applying all subdivisions) are

two methods for locating points within a section. Another method is the application of footages, which are two perpendicular distances, that are measured from a starting corner. These options are available to any user of LEO II who creates a proper data file of locations for entry into the conversion program (in batch mode) or who can enter appropriate values in response to program prompts (interactive mode). All of these options are discussed in this documentation.

The 35 rows of Kansas townships are numbered from T1S at the north to T35S at the southern border. The 'S' = south of the PLS base line (at the northern border of Kansas) may be omitted. The first township row begins at the PLS base line. In most township rows, there are 68 (east-west) columns of townships, 43 west and 25 east of the Sixth Principal Meridian (located at 108 miles west of the Missouri River on the base line). The range associated with a township, the "range designation," combines the range number and the direction to the east or west from the Sixth Principal Meridian, of the column containing the township. Range designations are from R43W (range 43 west), and run east to R1W (range 1 west), followed by R1E (range 1 east) and then east to R25E (range 25 east) at the state border.

Each township is identified uniquely by a township (row) number and a range designation - the row and column numbers of the township within the township grid. Sections define a six-by-six grid within (full) townships. They are numbered ("baustrophedonically") from 1 in the northeast corner to 6 in the northwest corner, and from 7 (below 6) to 12 (below 1), to 13 (below 12) to 18 and so on, until ending with section 36 in the southeast corner. This system was adopted in 1796 for unknown reasons (White, p.29).

b. Section Numbering Within Townships

NW NE

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

SW SE

Within each township, the sections are numbered in the usual ("baustrophedonic") manner from one (in the northeast corner of the six-by-six section grid) to six (in the northwest corner of the

grid) and from seven to twelve coming back from west to east in the next row of sections to the south. Section numbers increase to the west in odd rows, then drop down a row, and increase to the east in even rows. In the final, sourthernmost row, the first section (in the southwest corner) is number thirty-one and the last (in the southeast corner) is thirty-six, as shown below.

c. Section Subdivision

Point locations require a more precise definition than the areas defined by sections. Dividing sections into smaller and smaller areas is a systematic process that increases the precision of a location specification by a factor of two or four with each iteration. Sections are divided into sub-areas which in turn, are further subdivided, to a maximum of four levels, the limit in LEO II. Subdivisions are effected by passing from one to four specifications from the control routine to the conversion routine. The user does this by setting appropriate values for each level of subdivision in data records when creating the input data file for batch mode or by responding to prompts for user input in the interactive mode of the conversion program. LEO II expects the largest (which is applied to the section) subdivision first, and the smallest last. No subdivisions are required.

Sections and subsequent subdivided-areas may be subdivided into halves or quarters. Each is identified by naming the sub-area that is to become the next area for subdivision (or the final, "target area" of the process). Sections and sub-areas can be divided into halves by half-area subdivision using any of four possible halves: north, south, east, or west, such that choosing north results in the north half of the original area and so on. These half-areas are specified by a two-character value, comprising the first letter of the indicated direction (n, s, e, w) in upper or lower case, and a blank, represented as 'x'.

SPECIFYING HALF-AREA SUBDIVISIONS

NORTH = **N**x or n**x**

WEST = **W**x or w**x** EAST = **E**x or e**x**

SOUTH = **S**x or s**x**

Quarter-area subdivision divides a section or subdivided-area into four sub-areas and results in using one of these quarter-areas. Selection is made via either of two coding schemes. The most popular scheme codes the selected quarter-area by the applicable two-letter combination that indicates the direction of the quarter-area, ('nw,' 'ne', 'sw', and 'se') in upper or lower case. In the other scheme, quarter-areas are specified by single letters ('A', 'B', 'C', or 'D') in upper or lower case. The two schemes translate into equivalent specifications ('x' represents a blank):

Following quarter-area subdivision, the selected quarter-area becomes the next area of interest, and is further subdivided, unless it is the target area. Generally, many legal descriptions include at most three levels of subdivision. The operational limits of LEO II allow a maximum of four successive levels of subdivision, and a minimum of zero (no subdivision). Half- and quarter-

area subdivisions may be mixed together in any order, up to the limit of four levels. Non-blank specifications other than those listed are reported as errors, or may be ignored.

SPECIFYING QUARTER-AREA SUBDIVISIONS

NORTHWEST	NORTHEAST
NW or nw or Bx or bx	NE or ne or Ax or ax
SW or sw or Cx or cx	SE or se or Dx or dx
SOUTHWEST	SOUTHEAST

With any type of subdivision, all unspecified subdivisions should be blank (xx) or LEO II will attempt to interpret and apply them as additional subdivisions. If no subdivisions are specified (all are blank), the section itself is the target area, and is the basis for further specifications (only the closest point option could be used at that point). When a blank subdivision specification is encountered, LEO II assumes it is the last subdivision, and ignores any other specifications.

According to usual practice, a legal definition comprising the specifications ' Cx QA Hx QB ' would be interpreted (for a point location) as: "the center of the QA quarter of the H half of the QB quarter." Legal definitions commonly list the sub-area specifications from smallest to largest, as 'A of B of C'. To locate in geographic reference the point that was specified in legal reference in the example, the locations of the four section corners must be known. The largest (last) subdivision, 'QB,' is applied to the original section to locate the four corners of the QB quarter-area. From the result, the H half of that quadrilateral is determined, followed by the QA quarter-area of that half-area. The result is the set of four corner locations of the QA quarter of the H half of the QB quarter of the section. This final sub-area, determined by the sequentioal application of all subdivisions, is the "target area". The center location of this area, calculated by interpolation, or averaging the four corners, is the default conversion location.

In LEO II, the order of specifying subdivisions is reversed from the common 'A of B. . .' order, so that the order of subdivisions parallels the actual computations. The largest subdivision is applied first, to the section, and the smallest is applied last. LEO II expects the specifications to be in reverse order from what is commonly used. In LEO II, the first subdivision is applied to the section and each subsequent subdivision is applied to the result of the previous subdivision. So the sequence of subdivision specifications above would be interpreted in the opposite way: "take the QA quarter of the section, then take the H half of that, and then take the QB quarter of the result. Then, find the center of the final target area ." The center point of the target area is the default point location to be used in the conversion from legal to geographic reference, unless another is specified, such as with the closest point option.

Because LEO II is concerned with point locations, a single point must be selected from within the target area for conversion to geographic reference. This may be done by setting the closest

point indicator, with a value corresponding to one of the subdivision specifications or by leaving it blank for the default (it may be necessary to set this blank). To select the midpoint of one of the target area sides, the indicator is set to 'Sx', 'Nx', 'Ex', or 'Wx,' which selects the south, the north, the east, or the west side midpoint location. To select one of the corner locations from the target area, a quarter-area specification is used. The closest point indicator is set to 'NW,' 'NE,' 'SE,' or 'SW,' selecting the northwest, northeast, southeast, or southwest corner location, respectively. With the nine possible points in the target area, the closest area indicator can be used to reduce the maximum error in locating points by a factor of about two. After four levels of quarter-area subdivision, the target area is 330 feet square. Assuming the default center, the maximum error expected in locating a point is 230 feet, the average error is about 200 feet.

d. Footages

Point locations within a section can also be identified by using the "footages" option, in which locations are specified as being two given distances from the implied section corner within the specified section. Footages are two (orthogonal) distances that apply from a specified or implied section corner. Footages are used as a pirate might describe the path to buried treasure: "walk 1,200 feet north and 2,000 feet west, starting from the southeast corner." The footage method of specifying locations within a selected section has been used for a long time, and is used today in Kansas by oil companies and others working in the field. Application techniques for footages may have been sufficiently well defined for state surveys (White, c.1982), but they are not well-defined for LEO II. With no clear resolution of the issues involved, LEO II was forced to make some simplifying assumptions. These assumptions do not presume any uniformity in the size or shape of Kansas sections. One survey of Kansas sections (Collins and Wong, 1991) showed that real world deviation from original LEO assumptions is extensive, thus complicating the process of applying footages accurately. The LEO II conversion routine models the section geometry by projecting the corners of the quadrilateral to approximate the lengths of the section sides.

The footages option can be used in either batch or interactive mode of the conversion program. Of the two footage values given as part of a legal description to be converted, one represents a distance, in feet on the ground, that is either north or south. The other footage value represents a distance in feet on the ground that is either east or west. Both footages are specified as single whole numbers (integers) which may be positive or negative. The sign of each footage implies the direction in which the footage is to be applied. One value represents a north footage (to the north from the south) when it is positive, and a south footage (in absolute value) when negative.

This is the "north-south" footage. The "east-west" footage represents an east (from west to east) footage when positive, and a west footage when negative. In all cases, the distance used when the footage is applied is the absolute value of the original footage value (the amount without the sign). The two footage directions combine to imply the starting section corner. The possible combinations of footages are given below, with the implied starting corner for each. When a + sign precedes a footage, the distance used is the positive, when a - sign precedes, the (negative) value is negated. This is equivalent to using the absolute value of the footage.

FOOTAGE DIRECTION & START CORNER

NORTH-SOUTH FOOTAGE	EAST-WEST FOOTAGE	STARTING CORNER
+ FTNS > 0 (to North)	+ FTEW > 0 (to East)	SW
+ FTNS > 0 (to N)	- FTEW < 0 (to West)	SE
- FTNS < 0 (to South)	+ FTEW > 0 (to E)	NW
- FTNS < 0 (to S)	- FTEW < 0 (to W)	NE

4. REFERENCE CONVERSION

The main purpose of LEO II is to convert point locations in Kansas that are expressed in either legal or geographic reference into the equivalent location in the other reference, and to do so as accurately as possible for as much of Kansas as possible. In order to make conversions, there must be mechanisms that can recognize location descriptions in either system and associate each location with a unique set of coordinates in the other system. Converting points between some reference systems can be accomplished by direct application of a set of mathematical functions, which map any location from either system into its equivalent location in the other.

Because the legal reference system is based on the PLSS, locations are not expressible in any mathematically continuous functions. And, legal locations can not be described systematically with acceptable accuracy. The best and only possibility for conversion between the legal and geographic reference systems was to find or construct a set of locations for which coordinates are known in both systems of reference. Then, any point location in one reference system could be approximated with a high degree of accuracy by using the appropriate neighboring points to estimate the desired location through interpolation. The accuracy of such a conversion process would be controlled in large part by the accuracy of the original set of "control points".

The capability to perform conversions between legal and geographic reference in Kansas became possible once the complete set of section corners (all township corners are also section corners) for Kansas had been digitized from the USGS 7.5' topographic, "quadrangle" maps for the KCD project at the KGS. Through a complicated sequence of specialty programs, the corner locations were extracted from the KCD databases; analyzed to find inherent, spatial characteristics; sorted vertically, and horizontally within recognized rows; and analyzed again to generate the first LEO spatially organized database. A number of structural changes and location updates were made to the original database to create the more valuable LEO II database. Using it, LEO II can obtain the geographic coordinates of the corners of any section and use them to locate precisely points or areas within the section.

LEO II CONVERSIONS

LEGAL REF.	<< - - - - - - <<	"G TO L"	<< - - - - - - <<	GEOGRAPHIC REF.
(TSHIP,RNGE,SECT. .)	>> - - - - - - >>	"L TO G"	>> - - - - - - >>	(LONGITUDE, LATITUDE)

PROJCT CONVERSIONS

GEOGRAPHIC REF.	>> - - - >> "PROJECT" >>- - - >>	PROJECTED REF.
	<< - - - << "DEPROJECT" <<- - - - <<	(X,Y AT MAP SCALE)

C. THE LEO II DATABASE

The LEO II system utilizes a "spatial" database, which resides in the "LEO2BASE" file and is about one megabyte (1MB) in size. It is a spatial database, because it contains a set of section corner locations in geographic reference (longitude, latitude), which are organized so that each section corner location (and those of its neighbors) may be accessed directly, given the legal description for the corner. The LEO II database is the key to reference conversion, because it represents a discrete mapping between the legal descriptions and geographic coordinates for a large set of discrete points that covers Kansas. Given a complete set of (reasonably) accurate geographic locations of all Kansas section corners, it is feasible to approximate the geographic location of any point, defined in legal reference, by applying simple geometry and trigonometry to the locations of the four corners of the section containing the point or area to be converted.

1. KCD SECTION CORNERS

In 1972, digitizing began at the KGS on the "KDB" project, now called the Kansas Cartographic Database (KCD). The KCD is not a single database of cartographic information, but a collection of more than two thousand separate databases, each of which contains cartographic information from a "county patch." A county patch is defined as the intersection of a USGS 7.5' quadrangle map and a single county, and is represented in the KCD by a unique map database. Quadrangle maps in Kansas usually have from one to four contiguous areas that are bounded by county lines and the quad map edge - the county patches. Hence, there may be one to four distinct databases representing data digitized from a single USGS quadrangle map. The cartographic information in each county patch database was originally digitized from the corresponding USGS quadrangle paper map, rather than the more stable mylar maps, by students at the KGS.

Initial digitizing (by the research staff) began with a pilot project to produce a color-proof map of the geology of the Lawrence East, Kansas quadrangle, followed by the geologic contacts for the Geologic Map of the Lawrence West, Kansas quadrangle, beginning in 1979. In both cases, the digitizing included basic features for the initial county patch cartographic databases forming the KCD, as well as the geologic contacts. The KCD features were digitized from a set of large-scale, topographic maps published by the U.S. Geological Survey. This was convenient not only because they are the most comprehensive and accurate source of map data for the state, but the reference system inherent in them made them easy to manipulate. The four corners of each map are known in both geographic reference and projected reference, as soon as they are digitized.

The USGS 7.5' quadrangle maps, at a scale of 1:24,000, were the primary source of data for construction of the entire KCD and some related projects. The USGS maps, which completely cover Kansas at a scale of 1:24,000, were the highest quality source of cartographic information

available for Kansas. From these maps, operators accurately digitized the locations of all section corners, producing pairs of projected coordinates in "digital" (computer) form for inclusion in the KCD. The 80,000+ section corner locations were represented as Cartesian coordinates in inches on a map, based on the projection of the map. That information, encoded in each map database, guided the convertion to geographic reference through "deprojection," the inverse of projection.

Extension, expansion, updating, editing and selective retrieval of data (for production of geologic maps and digital data sets) of the county patch databases in the KCD has continued after more than twelve years. The building, editing and use of KCD map data requires the use of the KGS automated cartography system, GIMMAP (Geodata Automated Management Map Analysis and Production). The KCD databases contain hydrology, highways, county seats, railroads and other features, in addition to the original township and range lines and section corners. The statewide cartographic information of the KCD and the GIMMAP automated cartography software support publication-quality mapping and electronic transfer of geologic data for government and private agencies throughout the state, just as they supported construction of the LEO database.

The student operators at the KGS used the GIMMAP system to capture or "digitize" high-quality USGS map data and process it through the procedures (and programs) of GIMMAP, culminating in clean, topologically structured, graphically corrected databases. The set of processed data in the cartographic databases becomes the data store for production of "publication-quality" maps.

Using GIMMAP software, KGS students built and edited about 2,400 topologically structured, (county patch) cartographic databases from the maps they digitized. Other (supervisory) staff used other parts of GIMMAP to derive secondary databases by selecting and retrieving features from a set of primary, KCD databases and merging selected map features via "reprojection," to a common system of reference. The result of these efforts might be a cartographic database for a county, the state of Kansas, or some other "project area". The resulting, "project database" was then used to annotate, update, display, analyze, select, and compile data for various purposes, including: the transfer of valuable map data to other agencies; creation of customized research maps for talks and papers; and production of high-quality, color maps for sale or distribution.

2. DATABASE CONSTRUCTION

The LEO II spatial database is called "LEO2BASE," and resides in a disk file of the same name. LEO2BASE originally was constructed from the KCD data in 1985, in a process that involved several steps. The first step was to extract all section corner locations from the 2,000+ (KCD) cartographic databases that cover Kansas. Section corners were extracted on the basis of their unique feature codes, the four-digit numbers assigned to distinguish feature types (Ross, 1988b). The resulting set contained about 90,000 locations, more than expected. Probably, this was due to extra corners on correction lines and to the duplication of corners on county lines, one copy to each county patch. In this process, corner locations were deprojected to longitude and latitude (geographic) coordinates, using the map projection and deprojection routines from the PROJCT library, a part of the GIMMAP system. The extracted corner locations were doubly sorted, by latitude, and then longitude within rows to determine spatial relationships.

Successful construction of the LEO II database depended on the completeness and quality of the section corner locations, which were taken from the KCD. The database construction software relied on previous analysis of the spatial nature of the legal system to sort corner locations, first into horizontal bands of approximately equal latitude. Next, they were sorted within bands, by longitude, making their order in the file reflect their east-west relationships on the ground.

The result of this double sort was a single file containing 200+ rows of Kansas section corner locations in geographic reference, ordered from north to south, in bands of latitude. The section corner locations were ordered from west to east within each artificially recognized row. Using this structure, a series of complex analysis programs extracted the appropriate locations of the section corners of Kansas for the LEO II database. A series of specialized programs determined which sections shared a common corner location, which was added in geographic reference, to the appropriate record in the LEO II database, point by point, section by section.

When the LEO II database was constructed, the true complexity of the supposed "grid" used in the legal system was unknown. In the typical model, all sections were a mile square and fitted together neatly, with only a handful of known, systematic offsets in some places. This was the working paradigm used during the construction of the LEO II database. Because the true nature of anomalous sections scattered about the state was not discovered until much later, a significant number of errors, generally involving incorrect assignment of corner locations to sections, were incorporated in the construction of the original LEO II database.

Additional errors in recognizing correct section corner locations were caused by faulty analysis algorithms. Errors also arose from mistaken interpretations of PLSS features on maps prior to their initial digitization. A common example of this was the confusion between section lines and old boundaries of Indian lands. Some of the errors identified in this documentation have already been corrected in the latest version of the database. Some grid anomalies, involving incomplete or nonstandard sections, have been marked in their section flag values to prevent any attempt at reference conversion. A distinct set of problem areas await research, software, modifications of the database, and resolutions to the problems of internal specification for nonstandard sections. When these are resolved, the entire state will be open to reference conversion.

3. CONTENT AND STRUCTURE

The LEO II database contains one record for each township in Kansas and some extra (filler) records to simplify the mapping from township designation to record in the database. Kansas townships form an approximate grid, with thirty-five (township) rows from north to south and sixty-eight (range) columns from west to east. Actually, a grid with 35 rows and 68 columns would overlap areas outside of the state, especially in the northeast, where the Missouri river is the state boundary. Determining the record number of the database record that contains corner locations for a particular township, requires only a simple function of the township and range numbers. Accessing section corners in that record is a simple function of the section number.

For most of this township grid, the information associated with a single township, comprising the 49 section corner locations in logitude and latitude, in a fixed order, and "flags," which are single values that indicate the type or state of the township or section. In each township record, there is one township flag and 36 section flags, one for each possible section. Generally, these flags indicate where section corner information is missing (or perhaps, never existed). For every township, there is a township record, but not all township records correspond to real townships. The grid of 35 rows and 68 columns provides a place for each township, and the regularity of the grid provides for easy calculation of record numbers, given the township and range numbers. But, to cover the townships in a regular fashion, the grid must overlap areas outside the state.

Usually, the flag values and corner locations of a township are stored in a single, unique record in the database, corresponding to the township's position in the grid. Some database records do not correspond to any real township, but correspond to areas inside the grid, but outside of the state boundary of Kansas. Records for these areas contain only bookkeeping information, kept in the township flag, but not section flags or corner locations.

Most grid areas lying outside the state are in the northeast, beyond the Missouri River, but some are in the northwest. The northernmost row of townships, T1S, extends from range 42W in the west only to range 20E in the east. Grid columns to the east of that do not correspond to real townships. The tenth row, T10S, begins at range 42W in the west and ends at range 25E in the east. All grid elements correspond to actual townships, but range 43W does not. South of row ten, every element in every grid row corresponds to a real townships. Every column from range 43W in the west to 25E in the east represents an existing Kansas township. And each township has a corresponding, unique township record in the database, containing its corner locations and flag values. All rows before ten have some areas that correspond to empty township records.

In each full township, the thirty-six sections are defined by a set of 49 section corners, arranged in an approximately rectangular shape on the Earth's surface. These corners comprise a grid or array of seven rows and seven columns. This seven-by-seven grid is the basis for storage of the section corners for every unique township record. LEO II arbitrarily numbers the rows in this grid from one to seven, beginning at the top or northern edge and counting down or south to the bottom or southern edge of the grid. Similarly, the columns are numbered from one at the left or west edge to seven at the right or east edge. The first row comprises the section corners on the northern edge of sections six, five, four, three, two, and one. The second row comprises the corners on the southern edge of the same sections, and corners of the northern edge of sections seven through twelve (from west to east). Columns run from one on the west side of a section to seven on the east side. The complete ordering is shown in chapter III, Programming Notes.

4. SECTION AND TOWNSHIP FLAGS

In the original LEO system, the "township records" (so named because each record contains the section corner locations for a complete township) comprised only the geographic locations of the (township and) section corners in the township. With the subsequent discovery of various kinds of anomalies in the PLS (legal) system in Kansas, it became necessary to create a mechanism for

marking and recognizing nonstandard sections, where LEO II can not perform conversions. So, a set of section flags were incorporated into the database. Each section flag is a single integer value identifying the status of one of the thirty-six sections in the township. Most section flags have a value of zero, indicating a normal or standard section whose boundary can be defined by the four corner points in the database and, within which, the subdivision and footage application rules apply. Other values indicate a nonstandard section, where conversion is undefined.

The main purpose for adding section flags to the LEO II database was to identify and mark all sections in which the standard form of subdivision and other intra-section specifications, such as footages can not be performed. A primary example is the existence of sections with fewer than four defined corners, such as the northeast, where the Missouri River is the state boundary. In this area, many sections have partially irregular boundaries where the river is the state boundary. These sections are bounded by complex polygons, not quadrilaterals. The section flags for these sections have a value equal to the negative of the number of missing corners. The section flag for a section with three known corners would equal minus one (-1) - there is one missing corner. A section with one defined corner would have a section flag value of negative three (-3).

More generally, the reason for adding section flags was to have a way to identify and mark the "nonstandard sections", those areas in which standard subdivision and other specifications can not be used. The incomplete sections, those with fewer than four defined corners, are but one example of this phenomenon. Some other examples include sections that are:

(1) defined by six corners because they bend at the quarter-corners, apparently due to deliberate corrections at the local level;

(2) defined by five corners where multiple locations exist for a single corner, in turn, because survey lines did not meet;

(3) divided or distorted by a barrier (a hydrologic feature or government or private lands that were excluded from the PLS system) and which are all bounded by complex polygons comprising five or more points to define; and

(4) bounded by a five-point polygon resulting from the creation of a local correction line within the township, which resulted in non-matching lines that created multiple corners.

5. ACCURACY AND PRECISION

Several factors affect the accuracy of locations in the KCD and LEO II databases. More than 80,000 section corners in Kansas were digitized for the KCD. These were captured from USGS 7.5' (paper) topographic maps, which cover Kansas at a scale of 1:24,000. Although paper is not a stable medium and does contribute to errors, these topographic maps were (and still are)

considered the best available source of cartographic data for Kansas[7], and tests by the USGS on data digitized (for the KCD) from their paper maps showed a high degree of accuracy.

There are places where LEO II will not attempt a conversion, but for the whole of the state of Kansas, where LEO II does do conversion, the result generally is as accurate as any traditional source of such information. Obviously, the newer satellite technology (GPS) produces the most accurate results in today's environment. But using GPS technology requires putting a person in the field for every conversion. The principle factor in the accuracy of the LEO II conversions is the verified, high accuracy of the section corner locations, which were taken from the KCD.

In LEO II, sections can be subdivided into quarters or halves as many as four successive times. If all four levels are used, the final geographic location is calculated from a sub area of 330 feet square. If the legal location has been refined using the closest point option in the "target area," (the closest of nine different geometrically defined points) the maximum error may be reduced to 100 feet or less. The Global Positioning Satellite (GPS) network could produce higher accuracy, but a person would be required to locate section corners. The importance of the GPS system to the future of geographic location has been compared (Wilford, 1981) to the invention, by John Harrison of an accurate clock (1735), forerunner of the marine chronometer, which in turn, made it possible to determine longitude at sea within a half-degree.

The accuracy of the LEO II database could be improved with a comprehensive program to check and correct corner locations. This has not been done to date, because of the lack of funding and personnel. There may be other sources, such as the U.S. Geological Survey, who will provide superior data that can be incorporated in LEO II.

D. THE LEO II SOFTWARE

LEO II is the second generation of a software system developed at the KGS to convert location references accurately between legal descriptions (township, range, section . .) and equivalent geographic coordinates (longitude & latitude) for points in Kansas. Written in FORTRAN 77 for the PC environment, the LEO II system consists of two programs and a medium-sized (1MB) database (LEO2BASE) of section corner locations in geographic reference. The LEO II system grew out of the original LEO software and database, which was developed in 1985 at the KGS. LEO II represents significant changes in the content of the database, attendant modifications to the software, and a number of useful extensions to the conversion capabilities of the software. The original LEO II system was built and designed for the Data General minicomputer at the

[7] Digitizing for the KCD began in 1981, using paper maps that were available because the KGS was a vendor of USGS products. Much later, around 1990, after testing the quality of the KCD data against their own (unattainable) standards, the USGS offered a contract for KGS to map data from the mylar (stable-base) maps they would provide. In 1981, they wanted about $50 each for the maps. If they had offered free maps then, both would have been the richer. Ironically, our tests showed that the mylar maps were not particularly better than the paper ones, and also had comparable problems passing the USGS standards.

KGS. The new version of LEO II will not be supported on the minicomputer; it is exclusively designed for and maintained on the PC, with or without a math coprocessor.

1. CONVERSION AND QUERY PROGRAMS

The principal LEO II program, called LEO2PC ("lee oh - two - pee cee"), will convert locations in legal definitions or geographic coordinates into the equivalent location in the other system of reference. LEO2PC is a **reference conversion program** for people who have files of data in which additional information is associated with locations in Kansas, or people who create files of data with locations in Kansas, referenced in legal or geographic reference. Given a point or area within a specified section and township, LEO2PC will try to determine the longitude and latitude corresponding (equivalent) to the legal location. Within the section, points or areas are specified more precisely using various methods, such as subdivision and footages.

The conversion program converts locations from legal to geographic or from geographic to legal reference (the latter is a much less commonly used option). When the user has only a few point locations to convert, the LEO2PC program can operate in "interactive" mode. In this case, users must supply the input location coordinates by typing them at the keyboard, in direct response to on-screen prompts by the program. The equivalent locations, converted to the "other" reference system, are reported immediately and directly to the user, on the terminal screen. In interactive mode, the user can access the entire set of conversion capabilities of LEO II. This is not true in the non-interactive or "batch" mode of LEO2PC, unless the user has some programming support to modify the software. Only by modifying software can users access the "extended" functions and options outlined below.

The spatial database, LEO2BASE, containing the locations of all Kansas section and township corners in geographic reference is of fundamental important to LEO2PC. As described above, these locations were originally digitized as part of the Kansas Cartographic Database, comprising a set in excess of two thousand (GIMMAP) map databases containing cartographic information across Kansas. The KCD has been created and maintained at the KGS. Section corner locations for the KCD were captured in projected reference through digitization of USGS source maps and converted to geographic coordinates (by PROJCT software) for storage in the LEO II database.

In the LEO II system, there is a second program, called LEO2Q, that allows the user to examine section corner locations and township and section flag values in the database. With the LEO2Q program, it is possible to modify any of these values, if it becomes necessary. This program is of limited value to anyone other than those with programming support. On occasion, it may be necessary to learn the geographic location of a section corner, but modifications are almost never made. Necessary corrections will be made by the KGS, on a master database, which would then be copied and distributed. All errors found in LEO2BASE should be reported to the KGS.

2. FUNCTIONS AND EXTENDED OPTIONS

The basic functions of the LEO II system are to convert point locations from legal to geographic reference and vice-versa, whether the data are entered at the terminal or read from a disk file. The full capabilities of the system are normally available only in interactive mode, but extended functions are available to programming users. In batch mode, only a subset of the functions and options is normally available. In batch mode, data must comply with standard formats for legal or geographic coordinates and specification of program options (standard formats are defined in the program operation section). The listed functions are available in batch or interactive mode.

a. LEO II Principal Functions

1. convert a legal description with half- or quarter-area subdivisions and an optional closest point into geographic reference, longitude and latitude in decimal degrees

2. convert a legal description with east-west and north-south footages into geographic reference, in decimal degrees of longitude and latitude

3. convert a location in geographic reference (decimal degrees) into full legal description with four levels of quarter-area (nw, sw, se, ne) subdivision and closest point indicator

In interactive mode, the user has access to the full range of capabilities of the LEO2PC program. In addition to those functions available to non-programming, batch users (see previous list), the interactive user and the batch user who has access to some FORTRAN programming expertise may use the extended options and functions listed:

b. LEO II Extended Options

1. convert any type of legal description and receive the geographic location in degrees, minutes, and seconds (the "DMS" option) of longitude and latitude

2. provide a legal description of a section and receive the locations of all four corners in geographic reference, in decimal degrees or DMS

3. provide a legal description of a section and receive the approximate length of the four sides of the section in feet on the ground

4. specify a geographic location to be converted to legal description in DMS instead of decimal degrees

II. PROGRAM OPERATION

The LEO2PC (reference conversion) program is used to convert between legal and geographic reference, operating either in batch mode or interactively on a PC. The second LEO II program, LEO2Q (database query), supports examination and modification of the LEO II database. It can display corner locations for any section, from the LEO2BASE database. LEO2Q also allows the user to change the locations of selected section corners in the database. Although the ability to modify corner locations is essential, the need for doing so arises infrequently. When and if, changes become necessary, LEO2Q can display the original section corner locations, which can be changed directly and immediately. The LEO2Q program is intended for use by programmers to maintain and update the LEO II database. Operation of the LEO2Q program is described in the "Programming Notes" chapter. The current chapter concerns the operation of the LEO2PC reference conversion program.

A. REFERENCE CONVERSION

The LEO2PC program comprises four routines, a control routine (MAIN or LEO2PC), which interacts with the user, opens and closes files, reads input locations, sends input locations to the conversion routine, receives converted locations and outputs the converted locations. LEO2CVT, the conversion routine, is by far the largest[8]. Its function is to receive a location in geographic or legal reference, convert it to the other reference system, and return the converted coordinates to the control routine. If error conditions occur, LEO2CVT will note the type of error and return a status value that identifies the error. To do this, it must have access to the corner locations for all Kansas sections, which are in the database. For some conversion functions, LEO2CVT must use LEO2SETP and LEO2PRO to initialize polyconic map projection and to project geographic locations into a preset projected reference system, to approximate section side lengths.

1. OVERVIEW

The main function of the conversion program is to convert point locations from one system of reference (legal or geographic) into another, producing appropriate coordinates in the "output" reference system that represent the (approximately) identical location as that represented by the coordinates of the "input" reference system. In addition to its principal reference conversion function, the LEO2PC program performs two other, related functions: (1) return the locations, in geographic reference, of the four corners of any section, and (2) return the approximate lengths,

[8]The size of a program or routine can be measured by the length, in characters, of the source file, or the amount of memory allocated for execution. LEO2CVT greatly exceeds the other routines in the conversion program in both categories.

measured in feet on the ground, of the four sides of any selected section. Both of these options, and optional forms for input and output data, are available only in extended operations, which require program changes. In interactive mode, geographic locations may be expressed in an alternate form. The standard is decimal degrees, real numbers representing whole and fractional degrees. The alternate form is DMS (degrees, minutes, seconds), in which two integer values represent degrees and minutes, and a third real value represents whole and fractional seconds. This option requires program modification.

2. MODES OF OPERATION

The conversion program can be run (executed) in two ways. When only one, or a few point locations are to be converted, the program can operate in "interactive" mode, in which users can enter input locations at the keyboard of the terminal and receive output locations on the screen of the terminal. When many locations are to be converted, the user must create a file of input locations on the disk and specify the name of a second file for the output locations. The input file contains the input location coordinates, which follow the specific, fixed form of input for LEO2PC. This form of operation is "batch mode," which is the usual form for operating the conversion program when many locations, stored in disk files, are to be converted.

In batch mode, interaction with the user is limited to providing fundamental information, at the beginning of the operation. All input to LEO2PC must follow the required and optional forms, as presented in the introductory material, for defining locations in legal or geographic reference. The complete form for input data is fixed in batch mode, but not in interactive operation. Batch input locations are described in a disk file according to a fixed format, and the output location coordinates are written to a second disk file, also according to a fixed format.

In interactive mode, the user enters all information required for each conversion, one conversion at a time. The format for input coordinates is essentially "free form." The program prompts the user for operational choices and input coordinates, and checks each part on the fly. The user is prompted for all parts of the input coordinates in interactive mode. In batch mode, these values exist in the input file; only the file names, the data source (legal or geographic), and operational choices must be entered. In interactive mode, output coordinates are given directly to the user.

In the input file for batch mode, the precise location (in the file) of each part of legal definitions or geographic locations, as well as the content, magnitude, and type of each location part; and the operational choices must fit within acceptable ranges or the conversion will not be performed successfully. Values entered in interactive mode are not checked for error to provide realistic output. The fixed batch input and output formats, which can be changed by a programmer, are described in Chapter III, Programming Notes.

3. INITIAL CHOICES

Depending on the mode, the user must set values for appropriate options that control operation of the conversion program. In interactive mode, the options can be reset at any time. In batch

mode, they are fixed for each separate run. The user first selects the mode of operation, either "disk files," which is "batch mode" or interactive mode. The program prompts the user to enter the appropriate number for the desired mode. In either mode, the user must identify the input or "source" reference system (legal or geographic), by setting a value in each batch input record or entering the value at the interactive prompt. Output reference is the opposite of the source.

Next, the conversion program, operating in either mode, opens the LEO II database and checks for the presence of version information. The date of the current version of the database is stored (although not in earlier versions of LEO or LEO II) in an otherwise unused part of the database. LEO2PC will only operate with the current version of the database, which is included with the current version of the software. If an error occurs in this checking process, the user may have to replace the database with the current version. One other unusual event may occur in opening the database. When the version date first was placed in the LEO II database, the year was stored as a (negative) four-digit number. This resulted in unforeseen errors of small consequence, because LEO II programs that check, expect the flag values to be three digits at most, plus a minus sign, if one is present. The current version of the software checks for and corrects this condition.

After the database is opened, processing in interactive mode consists of defining a location in legal or geographic coordinates and viewing the output coordinates and the value of the status indicator, which indicates either a successful conversion or a specific error number. The user may convert any number of locations, exit, or reverse the conversion direction, at any time.

Once the database is opened in batch mode, the user is prompted for the name of the input data file, which is a complete pathname usually starting at the root of the working drive. Since the conversion operations occur from disk to disk and the user normally has no view of that process in action, the user can select to have the input data "echoed" at the terminal screen (=1) or not (=0). The program then prompts the user for the complete pathname of the output file for the converted locations. The user may wish to append (=1) or omit (=0) the status values (success or failure) and nonstandard section indicator values (some failures occur in sections that do not conform to the LEO II paradigm and are excluded from conversion) to each output record. The meaning of status and nonstandard section values are given in the introductory section.

4. INTERACTIVE CONVERSION

There are three parts to reference conversion using the interactive mode of LEO2PC. First, is to define the operating conditions; second is to define the input location coordinates according to LEO II rules; and third, is to view the results on the terminal screen. The first part has already been described. The definition of input locations is described here in separate parts for legal and geographic locations. This section covers the process of communicating location information. The meaning of legal and geographic location information is covered in the introductory chapter. Following the description of communicating input coordinates, is an explanation of the values that are output to the user's terminal screen.

In the simulated dialogs used in this section, the user-program interface is defined by example.

In the context of these dialogs, entries in *SMALL CAPS AND ITALIC* represent prompts, or output from a program, while those in **BOLD CAPS** are used for operator entries, which are only shown for unambiguous choices.

When the user enters a response, as indicated by the bold caps, it is assumed that the response is followed by a "newline," "carriage return," "enter," or equivalent character to signal the end of input to the computer. These characters do not appear in the dialogs.

[Comments and explanations are in brackets],. . . .and "/" separates options: "0/4" is "0 or 4."

OPERATING THE LEO2PC PROGRAM

To start the LEO2PC conversion program from the DOS prompt, the operator types

LEO2PC

ENTER MODE 0 = INTERACTIVE, 1 = DISK FILES? **0**

[Enter 0 for interactive mode ("Disk files" means batch).]

CONVERTING LEGAL COORDINATES

In interactive mode, the user is prompted for all input and enters simple free-format values.

DATA SOURCE OPTIONS
1 = LEGAL (TO LAT/LONG)
2 = LAT/LONG (TO LEGAL)
SELECT SOURCE OR ENTER 0 = STOP ? **1**

[Enter 1 for Legal source data.]

ENTER THE TOWNSHIP NUMBER? **TT**

[An integer (1-35), see introductory information above.]

ENTER THE RANGE NUMBER? **RR**

[An integer (1-43 West or 1-25 East), see introductory information, above.]

ENTER E(AST) OR W(EST)? **EW**

[A small or capital 'E' or 'W', identifying the range direction.]

ENTER SECTION NUMBER? **SS**

[An integer (1-36), as defined above.]

-2 = STOP PROGRAM
-1 = SELECT A SECTION OR RESET ALL OPTIONS
0/4 = SUBDIVIDE TO CENTER OR CLOSEST POINT
1/5 = SELECT SECTION/TOWNSHIP CORNER OR MIDPOINT
2/6 = APPLY FOOTAGES FROM SECTION CORNER
3/7 = RETURN LOCATIONS OF SECTION CORNERS
ENTER OPTION NUMBER (-2 TO 7) ? ***O***

[Selection of the OPTION value determines the process which is followed. From here, the user can branch to any of the possible operations. The program can be stopped with option -2. All options can be reset (option -1), including mode and source, or the section flag value and corner locations for a section, selected by number, can be displayed. The selected section can be subdivided one to four times (options 0 or 4), and the closest of nine standard points may be selected to make the location more precise. Footages may be applied from a section corner (options 2 or 6), or all four section corner locations can be retrieved (options 3 or 7). When OPTION > 3, longitudes and latitudes are each three real values, whole degrees and minutes, and decimal (whole and fractional) seconds. If OPTION < 4, each is one real value for decimal degrees. If footages (option 2 or 6) are zero, the program calculates approximate side lengths, in feet. After each conversion is completed successfully or not, program control returns to this selection of options. The user can continue with same section or enter -1 to change to a new area.]

ENTER CLOSEST POINT SPECIFICATION (CORNR)? **CP** [* Only for options 0, 1, 4, 5*]

[For options 0/4 and 1/5. When the target area is the section or subarea of the section, the location may be made more precise by selection of a closest point. There are nine natural options for the closest point - the four corners, the four side midpoints, and the (default) center of the area. These are specified by obvious two-letter identifiers: NW, NE, SW, SE, Nx, Sx, Wx, Ex, and Cx (or xx), where x = blank.]

FOOTAGES = 0, 0 FOR SECTION SIDE LENGTHS
ENTER FOOTAGE - NS, FOOTAGE - EW? **FTNS FTEW** [* Only for options 2 and 6 *]

[For options 2 and 6, two real values are entered for the "north-south" footage (FTNS) and the "east-west" footage (FTEW). A north-south footage is applied to the north when it is positive and to the south (in absolute value) if it is negative. An east-west footage is applied to the east when positive and west (in absolute value) if it is negative.]

CONVERSION OUTPUT

When options and location coordinates are specified, the program calls the conversion routine to attempt the conversion and displays the results of the operation on the terminal screen. The first thing displayed after each conversion is the status value, distinguishing successes from failures. Errors are identified by a status value. The program also displays the value of the "nonstandard section" indicator, which tells if the failure occurred in a nonstandard section. The remainder of program output after conversion of a legal location is the equivalent geographic coordinates, the longitude and latitude. This is the case for options 0/4, 1/5, and 2/6 (except for the side lengths option) with output in decimal degrees for options 0-2, and in DMS for options 4-6.

STATUS = X NONSTD = Y

[A status value (X) of 1 indicates success. All errors result in a negative status value. Status values are defined in Appendix A. The nonstandard section indicator value (Y) is set to 1 if the failed conversion was in a nonstandard section, and 0 otherwise. This part of the output occurs for LEO2PC conversions, but is not shown with other examples.]

When both footages are zero for options 2 and 6, the program finds approximate lengths of the section sides, in feet on the ground. Normally, this option is not available in batch mode, unless modifications are made to the program, as in the Extended functions. When the side lengths are selected, the output is displayed as shown:

LN (LENGTH OF NORTH SIDE)

LW (LENGTH OF WEST SIDE) *LE (LENGTH OF EAST SIDE)*

LS (LENGTH OF SOUTH SIDE)

If the north side of the section is approximately 5,122 feet, and the west side is approximately 5,284 feet, and so on. . . The output from the program might be:

5122

5284 5289

5176

Section corner locations are illustrated in spatial order as on the ground. The converted location is shown in the center:

LON, LAT (NW) *LON, LAT (NE)*

LON, LAT (CVT)

LON, LAT (SW) *LON, LAT (SE)*

The geographic coordinates can be returned for all of these options in either of two forms, the default "decimal degrees" form or the optional DMS form. With DMS, degrees, minutes, and seconds are kept separately, but with decimal degrees, minutes and seconds are converted to fractions of degrees and combined with the whole degrees. When decimal degrees are used, all returned locations are represented as two real numbers, the longitude and latitude. With DMS, longitude and latitude are both given as three numbers: degrees, minutes, and seconds.

CONVERTING GEOGRAPHIC COORDINATES

To convert from Geographic to Legal reference, select the second (lat/lon) option for the source:

DATA SOURCE OPTIONS
1 = LEGAL (TO LAT/LONG)
2 = LAT/LONG (TO LEGAL)
SELECT SOURCE OR ENTER 0 = STOP ? **2**

[Enter 2 for Geographic source data.]

Entering coordinates for locations in geographic reference is preceded by the selection of the form for the input geographic coordinates as either decimal degrees or DMS.

1 = DEG - MIN - SEC 0 = DECIMAL DEGREES
ENTER OPTION (-1 = RESET OPTIONS) ? **DMS**

[Selecting 0 or 1 sets the form of the geographic coordinates in the output. When -1 is entered, control will return to the beginning and allow the user to reset all options.]

ENTER LONGITUDE D, M, S ? **< lon: deg min sec >** [Only when form is 1.]

[For the DMS option, enter whole degrees from 0 to 179, whole minutes from 0 to 59, and whole and fractional seconds, from 0.0 to 59.999 - Kansas limits for DMS values are given in the introductory section. Do not use negative signs for longitudes.]

ENTER LATITUDE D, M, S ? **< lat: deg min sec >** [Only when form is 1.]

[As for the longitude, enter degrees, minutes and seconds for the latitude.]

ENTER LONGITUDE, LATITUDE ? **< lon > < lat >** [Only when form is 0.]

[For decimal degrees, enter real numbers, within Kansas limits. + or - longitude.]

CONVERSION OUTPUT

After the conversion is performed, the program will display the results on the terminal screen. The status and nonstandard values will be printed, followed by the complete legal description:

STATUS = XX NONSTD = YY

TOWNSHIP = TT
RANGE = RR EW
SECTION = SS
SUBDIVISIONS = S1 S2 S3 S4

[Four subdivisions and the closest point indicator are set. The first subdivision, s1, applies to the section to produce the largest subarea, and s2 is applied to that, and so on. The last subdivision, s4, produces the target area, in which the closest point selection is applied. The four subdivisions are expressed in the standard quarter-area (NW, SE,. . .) subdivision designations.]

CORNER = CP

[The corner or closest point option is included when the LEO II system creates a legal location. The closest point value is one of the corners, side midpoints, or center of the target area, whichever is closest to the location being converted. The closest point is specified by a quarter-area designation (NW,NE, . . .). It is blank (xx) for the center.]

5. BATCH CONVERSION

As with interactive conversion, there are three parts to reference conversion in the batch mode of LEO2PC. First, create the ASCII input data file according to LEO II rules for content and form; second, identify the input and output disk files and select the operating conditions in response to prompts; and third, examine the conversion results by typing, editing, or printing the output file. What the user must understand in order to do the first part has been described. Definition of the input locations according to LEO II protocol is described next, in separate parts, one for legal

input and for geographic input. This section describes the form and content of the LEO II input file, setting the operating conditions, identifying files, and viewing and interpreting the output.

ENTER MODE 0 = INTERACTIVE, 1 = DISK FILES ? **1**

[Enter 1 for batch. "Disk files" means batch.]

a. Converting Legal Coordinates

In batch mode conversion of legal descriptions to geographic reference, the input records include a switch to direct conversion program activity. The conversion program is able to determine the geographic coordinates of: (a) the center, side midpoints, or corners of a section or a township; (b) the center, midpoints, or corners of the "target area," which results from subdivision of the section, as described previously; or (c) the location that results from the application of footages. Half- or quarter-area subdivision of a section may be applied zero to four times. Two footages are given. The north-south footage is north when it is positive and south when negative. The other is east when positive and west when negative.

The first step in batch mode conversion is to create the proper input file. Then the user must indicate that the conversion source is legal (to geographic), identify the input and output files, and conversion begins, with output to the named file, in the fixed format.

INPUT FORMAT

In batch mode, the user must first create the input file(s). When converting legal coordinates, all records in the input file have the same form and content, but values in each field will differ from record to record. The input record for converting legal coordinates has some fields that contain location information and other fields with blanks for spacing:

x T T x R R x E W x S S x O x S 1 x S 2 x S 3 x S 4 x C P s F T N S s F T E W

Or with the fields grouped and separated:

x TT x RR x EW x SS x O x S1 x S2 x S3 x S4 x CP sFTNS sFTEW

Each "x" in the example record shown here represents a blank position or column. The actual spaces in the display records shown are not part of the data or the spacing of the record, they are included only for readability. The other fields encode the components of the legal definition, in the usual manner, but with some options or potential variations, as explained next. Ranges of acceptable values for each data field in the record are described in the introductory section. The fields shown above represent the values that are listed in the following table.

LEGAL INPUT RECORDS

COLUMN	FIELD NAME	MEANING
1	blank	
2,3	TT	Township number
4	blank	
5,6	RR	Range number
7	blank	
8	EW	Range Direction ('E' or 'W')
9	blank	
10,11	SS	Section number
12	blank	
13	O	Option number (0-7, as above)
14	blank	
15,16	S1	First subdivision, largest area
17	blank	
18,19	S2	Second subdiv. (applied to result of S1)
20	blank	
21,22	S3	Third subdivision
23	blank	
24,25	S4	Fourth subdivision
26	blank	
27,28	CP	Corner = Closest Point (of nine)
29-33	sFTNS	North or South (if s = -) footage * s
36-38	sFTEW	East or West (if s = -) footage * s

An example data record for the legal location: center of the east half of the northwest quarter of the southwest quarter of section 15 of township T22S, R34W:

```
column x 10                  1                   2                   3
column x 1  1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8
            x 2 2 x 3 4x w x 1 5 x 1 x s w x n w x e x x x x x x x x x x x x x x x x x
              ^^  ^^  ^  ^^  ^  ^^  ^^  ^^
             (TT  RR e/w SS  opt sub1 sub2 sub3)
```

A second example for the same township and section, but with footages (option = 3) of 2,222 feet north (positive, columns 29-33) and 1,234 feet west (negative, columns 34-38) from the (implied) southeast corner. When footages are used, subdivisions are ignored:

```
column x 10                  1                   2                   3
column x 1  1234567890123456789012345678901234578
            x22x34xwx15x3xswxnwxexxxxxxxxx2222-1234
```

The records in the input file for legal data will look more like the first example record, because the conversion program reads these records according to a fixed format. Each field value must be in the column(s) indicated in the table. One record must be created and entered in the input file for every legal location that is to be converted. An input file of these records can be created by manual or automated (computer) means. Small files can be created with the help of any text editor or wordprocessor. Large input data files usually are generated from other data files by special programs written for that purpose or by utilities such as a database management system (DBMS).

When the input file is ready, the conversion program can be run. First, the operating conditions or options must be selected:

DATA SOURCE OPTIONS

1 = LEGAL (TO LAT/LONG)

2 = LAT/LONG (TO LEGAL)

SELECT SOURCE OR ENTER 0 = STOP ? **1**

[Enter 1 for Legal source data.]

ENTER FILENAME FOR INPUT DATA? **< input file >**

[Enter the pathname (from the root directory) of the file containing the input data. If the file is in the current working directory, a simple filename is sufficient.]

ENTER 1 TO ECHO INPUT DATA, 0 IF NOT ? **< echo >**

[Enter 1 to monitor the progress of a conversion operation, by echoing data from the input records on the terminal screen. Otherwise, enter 0. The echoed data is not written to the output file.]

ENTER FILENAME FOR OUTPUT VALUES ? **< output file >**

[Enter the pathname (from the root directory) of the file to which the converted locations are to be written. If the file is in the current directory, the filename alone is sufficient.]

At this point, the LEO II database, LEO2BASE, is opened and checked for the presence of the correct release date in the first record, as explained previously. The program expects the LEO II database file, LEO2BASE, to be in a directory named DATA, which must be located at the root

of the default drive (for example, d:\DATA). Then, the program asks if the user wants to have the (conversion process) status information appended to each output record.

ENTER 0 TO OMIT, OR 1 TO INCLUDE THE STATUS AND NON-STD SECTION INDICATOR VALUES ? **< stat >**

> [Enter 1 to have the conversion status and nonstandard section indicator values added to each output record. The possible values for status are listed in Appendix A, but when the status is 1, if the conversion was successful, and it is negative when an error occurs. The user should check the output data for errors and make appropriate corrections. If there is an error and the nonstandard section indicator is not 0, the error resulted from attempting conversion in a nonstandard section, as described in the introductory section. To decline the status option, enter a 1 with the knowledge that errors may go undetected.]

The conversion process begins and continues until all records have been read from the input file, their locations converted to geographic reference, and the results written to the output file. If the echoing option is on, input records are copied to the screen, so the user can monitor the progress of the program. Output records contain geographic locations that are written to another disk file.

The standard LEO II form for the output in geographic coordinates is:

```
col x 10                    1
col x 1   1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
          - 1 0 0 . 1 2 3 4 5 x 3 8 . 1 2 3 4 5
```

In this example, the longitude is -100.12345 and the latitude is 38.12345. Five decimal places are used for longitude and latitude values written to the output file. Longitudes are written as negatives (west of the prime meridian) and latitudes are positive (north of the equator) values. The last digit of the longitude and first digit of the latitude are separated by a blank. In batch mode, geographic output can be sent to the output file in the DMS (degrees-minutes-seconds) form as one of the extended options. Access to all extended options requires a programmer to make minor changes to the program. This is described in the section on Program Extensions.

With either form of geographic output, the users have the option to have the conversion program append the status value and the value of the nonstandard section indicator to the output records. Both values are integers. The meaning of all possible status and nonstandard section values are listed in Appendix A. As mentioned, the status value is 1 for a successful conversion, and it is negative when an error occurs. The nonstandard section indicator is usually zero, but is set to 1 when a conversion fails in a nonstandard section (see introductory section). With these values (status = "stat", nonstandard = "ns") appended, the output records are eight columns longer:

```
col x 10                    1                   2
col x 1   1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7
          - 1 0 0 . 1 2 3 4 5 x 3 8 . 1 2 3 4 5 x s t a t x n s
```

If the conversion is successful, the status value will be 1 and the record may look like:

```
col x 10                    1                   2
col x 1   1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7
          - 1 0 0 . 1 2 3 4 5 x 3 8 . 1 2 3 4 5 x x x x 1 x x 0
```

If the conversion failed in of a nonstandard section, the status value would be negative (-99) to identify the specific error type. The nonstandard indicator value would be nonzero (1), and the geographic coordinates would not be correct, and should not be used.

```
col x 10                    1                   2
col x 1   1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7
          - 1 0 0 . 1 2 3 4 5 x 3 8 . 1 2 3 4 5 x x - 9 9 x x 1
```

b. Converting Geographic Coordinates

When the input or source data to a batch operation is in geographic reference, the user creates the input file, but with a format that is more flexible than for legal input data. Geographic input is read by the conversion program in "free format," meaning that the two values, longitude and latitude in decimal degrees, can appear in any columns as long as they are separated by one or more blanks or tabs, and the longitude appears first. The longitudes may be positive or negative. All geographic locations are accepted, but only those within Kansas will be converted to legal reference. The others result in errors. An example input record for geographic data might be:

```
col x 10                    1
col x 1   1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
          - 1 0 0 . 1 2 3 4 5 x 3 8 . 1 2 3 4 5
```

The same values may be entered in many other ways:

```
col x 10                    1                   2
col x 1   1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
          x x 1 0 0 . 1 2 3 4 5 x x x x x x 3 8 . 1 2 3 4 5
```

All characters other than the longitude and latitude must be blank. All digits for longitude and latitude must be consecutive, including the mandatory decimal point and the optional minus sign.

Given the input file, the conversion program is run and the options are selected:

DATA SOURCE OPTIONS

1 = LEGAL (TO LAT/LONG)

2 = LAT/LONG (TO LEGAL)

SELECT SOURCE OR ENTER 0 = STOP ? **2**

[Enter 2 for Geographic source data.]

ENTER FILENAME FOR INPUT DATA? **< input file >**

[Enter the pathname (from the root directory) of the file containing the input data. If the file is in the current working directory, a simple filename is sufficient.]

ENTER 1 TO ECHO INPUT DATA, 0 IF NOT ? **< echo >**

[Enter 1 to monitor the conversion operation, by echoing the input data records on the terminal screen. Otherwise, enter 0. The echoed data is not written to the output file.]

ENTER FILENAME FOR OUTPUT VALUES ? **< output file >**

[Enter the pathname (from the root directory) of the file to which the converted locations are to be written. If the file is in the current directory, the filename alone is sufficient.]

Next, the conversion program opens the LEO II database (\DATA\LEO2BASE) and verifies the version of the database by checking the release date in the first record, as explained previously, in the introductory chapter. Next, the program asks the user if the (conversion) status value and the nonstandard indicator value should be appended to the output records.

ENTER 0 TO OMIT, OR 1 TO INCLUDE THE STATUS AND NONSTANDARD SECTION INDICATOR VALUES ? **< stat >**

[Enter 1 to have the conversion status and nonstandard section indicator values added to each output record. The possible values for the status are shown in Appendix A, but the status is one (1) when a conversion is successful and negative when an error occurs. The user should check all output records for errors and make appropriate corrections. If there is an error and the nonstandard section indicator is not 0, then the error was attempting to convert in a nonstandard section, as described in the introductory section. To decline the status option, enter a 1, knowing that unrecognized errors may exist in the output.]

The conversion process begins, and continues until all records in the input file have been read, their locations converted to legal reference, and the results written optionally with the status and nonstandard section indicator values to the output file. If echoing was selected, each data record is copied to the screen, allowing the user to monitor the progress. Each output record contains a location converted to legal reference, in one of two fixed forms:

x TT x RR x D x SS x O x S1 x S2 x S3 x S4 x CP

without the status and nonstandard indicator, or with them:

x TT x RR x D x SS x O x S1 x S2 x S3 x S4 x CP x S T S x N S

The form of these records is almost identical to the format for legal input in batch, except that these records do not have footages, and the second form returns status and nonstandard indicator values (represented by variables 'STS' and 'NS'). Footages are not generated by the conversion program in this operation, because the labor of programming such an operation is greater than the value of the result, at this time. The user could generate them from the information provided or could use the system and this information to produce them indirectly.

This concludes the text for operation of the LEO2PC reference conversion program. The only other program that can be operated in the LEO II system is the LEO2Q (Query) program, which generally, is restricted to use by programmers or system managers. It can be used easily by any operator to locate quickly any section corner in geographic coordinates. Because of this useful function, its operation is discussed in brief.

B. DATABASE QUERY

The LEO2Q program is used to examine the corner locations of specified sections, with the option to modify these locations. In the process, the program also displays the township and section flags, values stored in each township record to identify partial and other nonstandard sections and townships with less than 36 section corner locations. Operation of the LEO II query program is described next.

As before, computer prompts and output are shown in *ITALIC, SMALL CAPITAL LETTERS.*

Entries by the operator are typed in **UPPER CASE BOLD,**

and comments or explanations are written within [square brackets].

The query program (LEO2Q) is a quick and easy way to locate section corners in geographic coordinates (longitude, latitude), by identifying a township and section, using legal coordinates as prompted. The query program operates only in interactive mode, not in batch. The query program can also be used to modify the content of the LEO II database, in \DATA\LEO2BASE, but this is operation not recommended for anyone other than systems staff, and then, only in rare instances. KGS updates of the database should make this unnecessary.

ENTER 1 FOR MODIFY MODE, 0 FOR QUERY ONLY ? **0**

[Enter 0 to examine section corner locations without modification. The modify option is described in the programming section.]

At this point, the LEO2Q program opens the database file (\DATA\LEO2BASE, on the default drive) and checks the version date as described earlier in this section.

ENTER THE TOWNSHIP NUMBER, (0 TO EXIT) ? **TT**

[Enter the township number for the township containing the section of interest. When all sections have been examined, the user enters a 0 here to exit the program.]

ENTER THE RANGE NUMBER ? **RR**

[Enter the range number of same. If the range number is less than 26, the program will prompt for the range direction (East or West), otherwise the next question is not asked.]

IS RANGE E(AST) OR W(EST) ? **EW**

[Enter the letter (e or w) of the appropriate choice (case is ignored).]

The LEO2Q program uses these selections to calculate the record number of the township in the database, and reads that record. The township and range numbers and township flag value form a header, and the geographic locations of the four township corners are displayed in their proper (spatial) relationship - as on a map - in geographic coordinates. The complete designation of the township is displayed above the corner locations, as: "T" followed by the township number and "S" for south, and the letter "R", followed by the range number and "E" or "W" for the direction (east or west). Displayed with the township designation is the value of the township flag, which is a number that may be useful to the programmer or database administrator.

***T** TSHIP# **S** **R** RANGE# **E/W** **FLAG** = TFLAG*

LON, LAT (NW) *LON, LAT (NE)*

LON, LAT (SW) *LON, LAT (SE)*

At this point, the two possible modes of operation, modify or query, follow different paths of operation. Use of the modify mode is described in the programming section.

SELECT SECTION (0 TO EXIT) ? **SS**

[Enter the section number of the section whose corner locations are of interest.]

The program locates the corners of the selected section and displays their geographic locations in their natural (spatial) relationships, with a heading that contains the township and range numbers, in addition to the section number and the value of the section flag:

T##S R##E/W SECTION ## FLAG = SFLAG

LON, LAT (NW) *LON, LAT (NE)*

LON, LAT (SW) *LON, LAT (SE)*

After this is information is displayed, the program prompts the user for another section number. The user may enter the numbers of additional sections in the township that are to be viewed or may enter 0 to select a different township (or to get to other options). At the township selection, the user selects a new township, or exits by entering zero.

III. PROGRAMMING NOTES

The original version of the LEO system did not have facilities for applying footages in a legal description. It also was incapable of recognizing sections that were "nonstandard" in one sense or another, such as those with nonquadrilateral boundaries that arose from hydrologic barriers encountered by the surveys. Attempts by the original LEO to convert in nonstandard sections produced results which were unreliable and almost certainly in error, or which might produce an error status, but without a meaningful message for the user. The current version of the system, LEO II, incorporates the ability to handle footages and to identify nonstandard sections, at least, most of the time. Additional changes also were made in the evolution to the second version of the LEO II system. Where they are significant to the user, these modifications will be described in this documentation.

The new options have been added in such a way that the primary system routines still operate as they did in the earlier versions, but now they support new capabilities. Adding several extended options has made the LEO II system more powerful, but not at the expense of compatibility with the original system. To accomplish these two design goals together, it was necessary to provide communication pathways for new information that is required for the extended features, without altering the original information pathways or interactions among LEO II routines. In the original system, information is passed through "arguments" or "parameters" associated with each "call" to a routine. It was necessary that program changes made for the extensions did not utilize routine parameters to communicate. Instead, they must communicate through a different FORTRAN 77 mechanism, the "common area." The parameters passed through LEO II routines and common areas are described in this chapter.

An added common area (the original system had one such area) where all new parameters are stored, provides the needed communication among LEO II routines to support the new extended options, while maintaining the original operating modes of the program. The new version can be used in the same way as the original has been used. By maintaining this compatibility, users of the old system who may have set up other programs or procedures to operate their conversions in a standard way will not have to make any changes unless they wish to gain the benefits of the new version and the extended functions and options. The price of this compatibility is that the conversion program must be modified to access the new parameters in the new common area.

A. SOFTWARE AND FILES

The LEO II system comprises a conversion program, LEO2PC; a spatial database, LEO2BASE, containing the geographic locations of all Kansas section corners and their spatial relationships to each other; and a database inspection and modification program, LEO2Q (Query) that may be used for a quick look at the locations of a few section corners. The structure, content, and use

of the database is discussed in the introductory section. The LEO II program components that are relevant to using the extended options, are outlined here.

The FORTRAN 77 source code for these routines is found in files with the same root name as the routines, and with an extension of .FOR (for FORTRAN). For example, the source code for the conversion routine is in the file named "LEO2CVT.FOR." Files with the compiled "object" code have names ending with the extension ".OBJ," and the executable file names all have the ".EXE" extension. Executable files contain the ready-to-run program. The program begins to execute or run when the root name is typed. For example, typing "LEO2PC" from the correct directory (or with an appropriate setting for the PATH variable), which contains the program in the LEO2PC.EXE file, will cause the conversion program to execute or "run."

The LEO II database, LEO2BASE, is in a file of the same name. Normally, the conversion and query programs expect this file to reside in a directory called DATA, which must be at the root of the default drive. If the DATA directory does not exist at the root of the default drive, it can created by typing: **MD \DATA**. Then, the LEO II database file, in the <sourcedir> directory, is copied into \DATA by typing: **COPY <sourcedir>\LEO2BASE \DATA**. The database file must reside in the \DATA directory, because the LEO II programs have a fixed pathname for the database file. This aspect can be changed so that the user specifies the pathname of the database file, for every run, by changing the OPEN statements of the programs.

LEO II SYSTEM FILES

FORTRAN 77 Source Code

LEO2PC.FOR	source code for the control (MAIN) routine, for the reference conversion program - LEO2PC
LEO2CVT.FOR	source code for the conversion routine for LEO2PC
LEO2SETP.FOR	source code for the utility routine to setup the map projection, for the conversion program
LEO2PRO.FOR	source code for the utility routine to use the projection to convert from geographic to projected reference (footages and side lengths)
LEO2Q.FOR	source code for the control (MAIN) routine for the database query and modify program - LEO2Q
LEO2FIX.FOR	source code for the utility routine for determining which neighbor townships must be also be changed, when updating the database
LEO2FIXR.FOR	source code for the utility routine that updates the section corner in a neighboring township as indicated by LEO2FIX

Object (Compiled From Source) Code

LEO2PC.OBJ	object (compiled) code for the control routine for LEO2PC
LEO2CVT.OBJ	object code for the conversion routine for LEO2PC
LEO2SETP.OBJ	object code for the projection setup routine for LEO2PC
LEO2PRO.OBJ	object code for the map projection routine for LEO2PC
LEO2Q.OBJ	object code for the control routine for the LEO2Q program
LEO2FIX.OBJ	object code for the corner update routine for LEO2Q
LEO2FIXR.OBJ	object code for the neighbor update routine for LEO2Q

Executable (Ready-to-Run) Code*

LEO2PC.EXE	executable file for the conversion program, initiate by typing LEO2PC from the appropriate directory
LEO2Q.EXE	executable file for the database query and modify program, initiate by typing LEO2Q from the appropriate directory

* Executable files are created by linking the associated object files via the system linker

LEO II Database*

LEO2BASE	LEO II database of section corners, must be under \DATA

* See chapters I and III for information on the database and the programs

B. PROGRAMS AND ROUTINES

The LEO2PC program comprises the following four (FORTRAN 77) routines:

LEO2PC - (or MAIN) the control routine that interacts with the user, opens files and the database, and calls the conversion routine to convert locations from one reference system to another. The only routine that must be modified to access the extended options.

Args: The control routine can have no arguments.

LEO2CVT - the conversion routine that accepts input coordinates that define a location, described in either the legal or geographic reference system, and which is to be

converted into coordinates in the other reference system. This routine also produces a status value to indicate if the operation was successful, if and which error occurred, and if the section is nonstandard.

Args: Arguments for LEO2CVT are described in "Passing Values."

LEO2SETP - the set-up routine that initializes the map projection software in LEO2PRO so that geographic locations can be projected to Cartesian (x,y) coordinates for applying footages or for estimating section side lengths. A particular form of the Modified Polyconic projection (the central meridian is located at the center of the section) is used at a scale of 1:12, with the result that output projected coordinates represent feet on the ground.

Args: There is one argument to the LEO2SETP routine. CORN is a double precision array of 9 elements, which returns the locations of the four corners of the map projection area, and the minimum y value for the map area (at center-bottom of the map area), in projected (map) inches. CORN (1) = xsw, (2) = ysw, (3) = xse, . . . , CORN (7) = xnw, and (8) = ynw. CORN (9) is the minimum y value for the projection area, a useful value for map display and design purposes.

LEO2PRO accepts a point longitude and latitude (in decimal degrees) and projects the point, using a particular form of the Modified Polyconic projection, at a scale of 1:12, to return the rectangular (Cartesian) or projected coordinates of the point with units equal to feet on the ground.

Args: There are four arguments for the LEO2PRO routine. The first two, PLOND and PLATD, are the longitude and latitude, in decimal degrees, of the point in Kansas, that is to be projected. The second arguments, XP and YP, are the projected coordinates in feet on the ground. All four arguments are double precision.

The LEO2Q program comprises three routines:

LEO2Q - (or MAIN) the control routine interacts with the user to select all options, identify townships and sections, and, if modify mode is selected, make changes in the database. This routine opens and accesses the database, reading township records to obtain and display township and section corner locations and flag values.

Args: No arguments.

FIX - a routine that is called by the control routine when modify mode is on and the user modifies a section or township corner location. If the modified point is on the edge of a township, then the equivalent section location in the neighboring township record must be updated. And when a township corner is changed, all three neighboring township records must be updated. The FIX routine analyzes and determines which other records must be updated individually, by FIXR.

Args: The arguments fo FIX define where, in the database, a change in the location of a section or township corner was made, and the coordinates of the changed location. The arguments identifying the changed record are:

RECNUM = the township record number containing the changed corner
TSHIP, RANGE = the corresponding legal description (only for a report)
ROW, COL = the row and column of the corner within the record

These arguments are all integer (2 bytes) variables, while the changed location is expressed in real values by the arguments, LON and LAT, which contain the new longitude and latitude for the corner, to be used for updating the equivalent corner in adjacent records.

FIXR - the routine that receives a township record number and information to identify a corner to be updated, along with the new location coordinates, and then reads the record, changes the corner location, and writes the record back to the database.

Args: Arguments to the FIXR routine describe the location, in the database, of a section corner that is equivalent to a modified corner (and must be updated), and the new location for the corner, in decimal degrees of longitude and latitude.

C. PASSING VALUES BY ARGUMENTS

For the conversion program to work, the control routine and the conversion routine must pass information to each other about location coordinates, user-selected options, status information, and the value of the nonstandard section indicator. Programming in the FORTRAN 77 language usually provides two methods for communicating information. One method is to pass values through "arguments," the second is through "common areas." The arguments for all routines but the main conversion routine, LEO2CVT, have been explained in the previous section. Those for the conversion routine are detailed in this section. The operations that involve passing values by common area variables are those associated with the extended functions. These are described in the Extended Options & Functions chapter.

In the argument method, each routine that "calls" or invokes another routine can attach a set of variable and constant values that are expected by the second routine. The second routine begins operation with this set of "arguments" available to its use, usually through a temporary storage area where the locations of the parameters are passed. With this method, the parameters sent to a called routine may differ from one call to the next. The argument or "parameter" list was used in the original version of LEO and is used in LEO II for the basic functions: legal/geographic conversion, subdivision, closest point, and footages.

In order for these arguments to be transferred correctly in either direction, both routines must be accessing and interpreting them in the same manner. This occurs only when the number, order, type, size, and logical content of these arguments are identical as used by the calling routine and

the called routine. The arguments used for the LEO2CVT routine are shown with names as used in the program, and in proper calling order. Descriptions of each parameter follow.

CALL LEO2CVT:

(TSHP, RANG, EW, SECT, OPT, SBD, PLON, PLAT, LUN, STAT, SORCE)

where:

TSHP = township number of a legal location, INT*2 (type = integer, 2 bytes)

RANG = range number of the legal location, INT*2

EW = east/west indicator for range number, CHAR*1 (type = character, 1 byte)

SECT = section number for the legal location, INT*2

OPT = user's option for conversion type, INT*2 (see Program Operation). Options include township/section corner, subdivision, closest point, and footages.

SBD = subdivision specifications, CHAR*2 (4), array (see Introduction) Quarter- or Half-area subdivisions, first applies to section, unused are blank.

PLON = longitude of geographic location in decimal degrees. Longitude in Kansas can be positive or negative, REAL*4 (type = floating point, 4 bytes).

PLAT = latitude of geographic location in decimal degrees, REAL*4

LUN = logical unit number for the already opened database file, INT*2

STAT = status of the operation (1=success), INT*2 (see Appendix A)

SORCE = source (1=legal, 2=geographic) of input data, INT*2

D. THE LEO II DATABASE

The LEO II database is the central focus of the LEO II conversion system, because it contains the geographic locations of all section and township corners, arranged in records by townships and within townships by spatial relationship within the townships. This arrangement makes it possible to perform reference system conversions from legal to geographic or geographic to legal reference anywhere in the state. The structure and content of the records in the LEO II database, access to the information in the database, and the meaning of additional values in the database, are the topics of this section.

1. TOWNSHIP RECORDS

The original LEO database contained one record per township. Township records corresponded to elements of the Kansas township grid, comprising thirty-five rows from north to south, and a maximum of sixty-eight columns per row, west to east. This model of townships in Kansas was used to design the LEO database. Each township record was located on the basis of its township (row) number from 1 to 35, and its range (column) number and direction. Rows were extended if necessary, to 68 columns, making it easy to determine township record numbers, which might otherwise be more difficult. In the northeast, where the Missouri river bounds the state, some of the rows of townships would have less than 68 columns if they ended at the state boundary. If rows ended at the river, some would have less than 68 columns (townships). With unequal row lengths, determining a township record number would require additional knowledge.

2. ACCESSING THE DATABASE

Before any conversions can be performed, the LEO II database must be opened with a statement that resembles:

```
      OPEN (LUN, FILE='\DATA\LEO2BASE', ACCESS='DIRECT', MODE='READ',
     +      FORM='UNFORMATTED', RECL=466)
```

To access a record in the database for a particular township or its sections, the correct record number must be calculated from the township, range, and east/west information:

$$TREC = 68 * (TSHIP - 1) + RANGE$$

where:

TSHIP = township number (1-35)
TREC = township record number

RANGE = 44 - West Range number, or
= 43 + East Range number

To read a township record (which must be done at least once) for a conversion (LUN = unit number of the file containing the database, probably \DATA\LEO2BASE):

```
READ (LUN, REC=TREC) ( (LON(I,J), LAT(I,J), J=1,7), I=1,7), TFLAG, SFLAG
```

3. STORAGE OF CORNER LOCATIONS

When the township record is read, the section corner locations are stored in variables LON(7,7) (longitudes) and LAT(7,7) (latitudes), both real*4 arrays of seven rows and seven columns. The order of storage for these section corner locations is shown below. In the diagram, the section numbers are centered within the sections and are bold and underlined, such as **24**. The pairs of numbers, each from 1 to 7, inside parentheses, are the row and column numbers of an element in

the LON and LAT arrays. Each corner longitude and latitude is stored in the 7x7 array just as it appears in the 7x7 array of section corners on the ground. The northwest corner of section **6** is in row one, column one and the northwest corner of section **5** is in row one, column two.

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

4. MISSING CORNERS

Each township record in the LEO II spatial database, LEO2BASE, contains values representing the geographic (longitude and latitude) locations of the forty-nine possible section corners that exist in a complete township. When townships have fewer than forty-nine known section corner locations, because of corners whose locations are unknown or to corners that do not exist, the LEO II system has marked the absent corner locations with special values. All unknown section corner locations in the database have as their location a positive longitude (9.9) and a negative latitude (-9.9), neither of which is possible in Kansas. The presence of these markers indicates that the section corner location is unknown or nonexistent. All LEO II conversion and database operations observe these values, and halt to report an error when these are encountered.

5. TOWNSHIP AND SECTION FLAGS

The database of the new LEO II system contains information about townships and sections that was not available in the original LEO system. Each township record now includes a set of 37 integer values, termed "flags," one associated with the township itself, and one associated with each section. These flags warn LEO II of special conditions for some sections in limited areas of Kansas, in which LEO II can not convert point locations

The first of these values, immediately following the section corner locations in the record, is the "township flag," whose value indicates the status of the overall township. It has a value of zero (0) if all 49 section corner locations in the township record are known and defined. For records that exist only as padding (for a rectangular grid) and correspond to nonexistent townships with no complete sections (a complete section has four defined corners), the township flag is minus one (-1). The township flag for township records with at least one, but fewer than 36 complete sections is equal to the number of complete sections in the township (1-35). The township flag is stored in the integer variable "TFLAG."

TOWNSHIP FLAG VALUES (TFLAG)

0	"complete" townships, with 49 defined corner locations
-1	empty townships, outside the state, with no corners
0 < N < 36	townships with N defined corners
N < -1 or N > 35	are illegal values with no (current) meaning

The remaining 36 values in the LEO II database records are termed "section flags." The value of each section flag identifies one of three possible conditions. If a section flag is zero (0), all four corners exist, their locations are known, and they are contained in the database. These are "standard" sections, with respect to LEO II subdivision, footages, and intra-section locations. The section flags are stored in the integer array "SFLAG."

SECTION FLAG VALUES (SFLAG)

0	"standard" sections with four defined corners
-4 < N < -1	sections with (-N, 1-4) missing corners
0 < N	nonstandard sections, N indicates class (currently unused)
N < -4	illegal section flag values with no meaning

When one or more section corners are missing or are non-existent, the section flag is set to the negative of the number of undefined corners in the section (-1 to -4). The section flag may be set to a positive value, if the section is nonstandard (if the boundary is not a quadrilateral or if subdivision within the section can not be performed in the usual way). Numerous sections are known to be nonstandard, but this group is only a very small part of the whole. To date, little work has been done to find methods for subdividing or applying other sub-section specifications for locations in these sections. Currently, the only sections with non-zero flag values are those that, usually due to hydrologic or other boundaries, have one or more corners that are undefined

or are unknown. Other classes of nonstandard sections have been identified, but for now, code numbers for their flags have not been established. In the future, such code numbers may locate specific information or procedures used to define locations within nonstandard sections.

E. ALGORITHMS

A number of useful algorithms are given to assist the programmer in understanding the workings of the conversion routine and to assist the programmer in accessing data in the LEO II database.

1. STANDARD SUBDIVISION

Standard subdivision is the straightforward way to divide into halves or quarters a quadrilateral. To halve an area for either the north or south half, the midpoints of the east and west sides are found by averaging the locations of the corresponding endpoints (northwest and southwst corners for the west side and the NE and SE corners for the east). These derived locations are combined with corner endpoints of either the north or south side to form the half-area. A parallel process is used to find an east or west half.

Finding the quarters of an area is similar, except that two lines are constructed, one between the midpoints of the east and west sides, and another between the midpoints of the north and south sides. These two lines intersect at the centroid, a central point of the area (equal to the average of all four corner locations). The desired quarter-area is defined by the original corner indicated in the subdivision specification, the two side midpoints of the original area closest to the corner, and the calculated center of the original quadrilateral.

The method for standard subdivision, as described, is acceptable for most sections for which all four corner locations are known. However, due to techniques used in the original survey, there is a set of sections for which a different subdivision method could produce more accurate results. Because the PLS survey started measuring sections in the southeast corner of each township, the sections laid out in the first five rows and columns to the north and west are nearer the intended (one mile square) dimensions than those sections in the northernmost row and the westernmost column of the townships. These areas were intended to absorb the errors and differences so that they would be whatever size was left.

The sections in the first five rows and columns west and north from at the southeast corner tend to be divided evenly by lines connecting their quarter-corners, as is done in standard subdivision. The quarter-corners of sections in the northernmost row and westernmost column of townships were surveyed, as for all sections, to be a half-mile from the southeast corner of the section. In these smaller or larger sections, division into "quarter-sections" can not be fairly represented by the even division process of standard subdivision. For these sections, the quarter-sections must be defined by vertical and horizontal constructed lines, each at one half-mile from the southeast corner of the section. The result is that the southeast quarter-section will approximate a quarter-mile on a side, but the other sections may be larger or smaller.

These techniques for subdividing the northern row and western column of sections in a different manner are preferred over standard subdivision as was used in the earlier versions of LEO II, but they are just being incorporated at the time of this writing. Following the printing of the LEO II documentation, a new version of the software, which incorporates these changes, should be made available for general distribution.

2. LOCATING SECTION CORNERS

For the programmer who wishes to add or modify code that involves section corner locations, which are stored in LON and LAT, it is helpful to have algorithms for converting back and forth between section numbers and row and column numbers of the section corners as they are stored in the two arrays. A previous diagram shows the arrangement of the section corners in the LON and LAT arrays. The algorithms for converting section numbers to the corresponding row and column numbers (and vice-versa) for the NW corner of the section are:

a. Section Number To Row & Column

```
SROW = ( ( SECT - 1 ) / 6 ) + 1
SCOL = MOD ( INT ( SECT), 6 )
IF ( SCOL .EQ. 0 ) SCOL = 6
IF ( MOD ( INT ( SROW ), 2 ) .EQ. 1 ) SCOL = 7 - SCOL
```

where:

SROW = row number of the NW corner of the section, INT*2
SCOL = column number of the NW corner (in LON, LAT), INT*2
SECT = section number, INT*2

NOTE: The NW corner of SECT is at LON (SROW, SCOL) and LAT (SROW, SCOL).

b. Row & Column To Section Number

```
IF ( MOD ( INT ( SROW), 2 ) .EQ. 0 ) THEN
    SECT = ( 6 * ( SROW - 1 ) ) + SCOL
ELSE
    SECT = ( ( 6 * SROW ) + 1 ) - SCOL
ENDIF
```

where:

SROW, SCOL, and SECT are as before.

NOTE: The longitude of the northwest corner of section SECT is LON (SROW, SCOL) and the latitude is LAT (SROW, SCOL).

3. DECIMAL DEGREES AND DMS

To convert between the two forms for geographic coordinates, decimal degrees and the DMS (degrees, minutes, and seconds) form, use these algorithms:

a. DMS To Decimal Degrees

```
DD = FLOAT ( WDEG ) + ( FLOAT ( WMIN ) / 60. ) + ( WFSEC / 3600. )
```

where:

```
    DD = decimal degrees, REAL*4
  WDEG = whole degrees, INT*2
  WMIN = whole minutes, INT*2
 WFSEC = whole & fractional seconds, REAL*4
```

b. Decimal Degrees To DMS

```
  WDEG = INT ( DD )                          [ integral part, (whole degrees) ]
  TEMP = (DD - WDEG) * 60.                   [ whole & fractional minutes ]
  WMIN = INT ( TEMP )                        [ integral minutes ]
 WFSEC = ( TEMP - FLOAT ( WMIN ) ) * 60.     [ decimal seconds ]
```

where:

```
    DD, WDEG, WMIN, and WFSEC are as before
  TEMP = temporary storage, REAL*4
```

F. MODIFYING THE DATABASE

As described in the Program Operation section, the LEO2Q (Q = Query) program will display the contents of the LEO II database, for any township that the user selects through responses to prompts by the program for legal reference coordinates. Normally, update mode is off, and the LEO2Q program is used only to examine the geographic locations of section corners. If update mode is on, which should be a rare occurrence, the LEO2Q program will display the township and section information, and will also allow the user to modify the content of the record that is displayed. When update mode is on, the program prompts the user for selective changes in the corner location or flag values.

Like the LEO2PC conversion program, LEO2Q expects to find the LEO II database in the file LEO2BASE, which resides in directory \DATA at the root of the default system drive. This is one of many aspects of the LEO II software that is open to changes by a programmer that might benefit users. Part of the operational dialog common to both the modify and no modify modes is given in the Program Operation section. The remainder of the dialog applies only to modify mode. Starting with selection of the section, this dialog is simulated next.

SELECT SECTION (0 = TSHIP, -1 = EXIT) ? **SS/0**

[To modify the township flag, enter 0. To modify a section flag or any of the section corner locations, enter the section number, 1-36.]

ENTER NEW TOWNSHIP FLAG VALUE ? **< T-flag >** [Only with section = 0 option.]

[Possible values to enter are described above. Illegal values are not accepted. After this, the program returns to the township display.]

If a section is selected by entering a value from one to thirty-six, then the user can modify the section flag or any corner location in the section, but only for sections whose section flags are zero in value (standard sections).]

ENTER POSITION TO MODIFY (NW,NE,SW,SE,F=FLAG) ? [Only with section = 1-36.]

[Select the corner position to modify using the direction to that corner, or choose to modify the section flag, whose value is also posted. Entering a null (newline) value or any value not in the list, returns control to the "section number" prompt above.]

TO DELETE A POINT, ENTER (1., -1.)
ENTER NEW LONGITUDE, LATITUDE? **< lon lat >** [Only if a section corner is chosen.]

[Enter the new longitude and latitude for the corner to be changed. If the location is to be represented as unknown or undefined, enter +1., -1. and the default location values for undefined corners are inserted (longitude = +9.9 and latitude = -9.9, both are impossible in Kansas). Any legitimate Kansas longitude and latitude may be entered, and the usual bounds apply: longitudes from -102.125 to -94.5 and latitudes from 36.875 to 40.125. The new values replace the old ones in the record, the record is updated in the database, and all bordering townships - those that share the updated corner - are updated as well. The display is updated, and the program returns to the "position to modify" prompt.]

ENTER NEW SECTION FLAG VALUE?

[Enter the value to which the section flag is to be reset. Any value is accepted since all possible values have meaning for section flags. Care should be taken that this value is correct or the LEO II system may not function properly. The meaning of these different section flag values are given above under "The LEO II Database - Township and Section Flags."]

IV. EXTENDED OPTIONS & FUNCTIONS

When the original LEO system was revised, changes were made to supplement the functionality of the system beyone the original primary functions, which were:

1. convert a legal description in Kansas to longitude and latitude

2. convert a geographic (longitude, latitude) location to an equivalent legal description

3. locate a specific section or township corner in geographic reference

4. report meaningful error information whenever a conversion failed

Changes were made to add footages (measured from a known section corner), to locate all four of a section's corners in geographic reference in one operation, to approximate the lengths of all four section sides in feet on the ground, and to provide for entry and output of longitudes and latitudes (geographic reference) in an alternate, but commonly used form (separate values for degrees, minutes, and seconds), instead of the usual "decimal degrees" form.

These changes were made without changing the original functioning of the conversion software, so that existing users could continue their operations without making any changes. These new functions and options were implemented without affecting the original functionality of the LEO (now LEO II) system. This was accomplished by means of tools available in the FORTRAN 77 programming language, but not without a price. In order to access the extended functions, users must modify the control routine of the conversion program, LEO2PC as will be described. Only the footages option is available to all users. The others require the intervention of a programmer to make small changes.

A. PROGRAMMED EXTENSIONS

There are three basic extensions to the functionality of the conversion program. Two of these are functions that will produce results that could not be obtained from the original version of the system. Even with the new version of LEO II, which incorporates footages and other changes, these two functions are not accessible without making program changes. The third new aspect of changes for LEO II is an option that allows users to define geographic coordinates (longitude and latitude) in an alternate form, other than "decimal degrees," which is the default or standard form in LEO II. This optional form can be used to enter geographic locations for conversion or it can be applied to geographic output from the conversion program. These expanded functions and options, described generally in the following three blocks of text, require the existence of the new common area to communicate values between the main routine and conversion routines.

1. DMS

"Degrees, Minutes, and Seconds" (DMS) is the alternate form for specifying geographic coordinates for input or output. Originally, all longitudes and latitudes were expressed in "decimal degrees" - each a real number that comprises the whole and fractional parts of degrees (for the minutes and seconds). In the DMS form, these coordinates can be given in three parts - whole degrees, whole minutes, and decimal seconds - the latter combines the whole and fractional seconds into a single real value. The DMS option can be used in almost any LEO II operation, in place of the standard decimal degrees input or output. DMS is selected by specifying the appropriate OPTION value. An method for converting between decimal degrees and DMS is given in the Programming Notes chapter.

2. FOUR CORNERS

Some applications that use LEO II conversion might require the section corner locations only. Sometimes, they might need to obtain the geographic locations of the four corners of a section. This was possible with the original program, but it required four separate calls to the conversion routine. With the Four Corners option, the user can receive the locations of the corners of a selected section in one operation. Similar to the other extended options, the four corners option requires access to the new common area, and hence, requires program modifications, which are described in this chapter. As with the other options, the four corners option can produce geographic coordinates of the four section corner locations as decimal degrees or in the DMS form.

3. SIDE LENGTHS

The LEO II conversion routine (LEO2CVT) requires the ability to determine the lengths of the section sides for any section, given its legal description, in order to be able to apply footages used in legal locations. Calculating approximate section side lengths is accomplished by two routines that utilize a Modified Polyconic map projection to convert the geographic coordinates of section corners into Cartesian coordinates. With those two routines, it is easy to make available these useful values to users of the system. LEO II users can request and receive the APPROXIMATE side lengths of any section specified by a legal description. The side lengths, given in feet on the ground, are subject to many sources that may induce error. They should be used only for rough approximations, until their confidence is thoroughly checked. As with other extended options, the side lengths option uses the new common area to pass values. The side lengths option applies only to WHOLE sections; subdivisions are ignored, if present. The section side lengths option is invoked by choosing the footages option with both footages = zero (0). The side lengths are placed in the common area in the order: North, East, West, and South (N-E-W-S).

These three extended options can be used only by making small changes in the MAIN (control) routine of the LEO2PC program (The control routine, MAIN, might be named LEO2PC, resides in a named MAIN.FOR or LEO2PC.FOR). Making other changes in the original FORTRAN 77

source code of the conversion program can alter the program to read local data containing legal or geographic locations to be converted. The input format can be set to accept whatever data the user has, and the output data, with converted locations and the other original data, can be written in any form to the output file. Thus, the program can be customized to read and write local data files, and convert the reference from the input data to the output data. With program changes, data with local attribute or location information can be properly maintained for local operations.

B. PASSING VALUES BY COMMON AREA

Implementation of the extended functions in the second version of LEO II created the need for a different method to communicate values between LEO II routines. Because a major design goal was to make the new version compatible with the first - that is, users of the original LEO system would be able to use the same calls and input format as before. For them, only the name of the conversion routine and the database had to be changed. If an original user had created and was using a customized version of the conversion program, then the initialization part and calls to the conversion routine could remain the same except for the "2" in the routine name, "LEO2CVT."

The second form for sharing data between routines does not allow passing values of different variables in different calls to the conversion routine. The COMMON area approach to routine communication provides for fixed areas in memory for what are essentially "global" variables, which are variables that are accessible to all routines that include the appropriate (COMMON) declaration for the reserved area. The COMMON area used to communicate values between the LEO2CVT and LEO2PC routines is the "labeled" COMMON area, because such areas are given names or labels, by which they may be referenced. Each uniquely named COMMON area is associated with a set of variables, which usually must be identical in number, order, type, size, and logical content for all routines which refer to the area.

In LEO II, the labeled COMMON area is named "LEOXTR" (for LEO extra values). It contains variables that are used primarily for extended options, but it also contains variables that are used for the original LEO functions. Though this set of tasks seem the same in the two versions from the user's point of view, some tasks have been implemented differently in LEO II. Some tasks now use the new LEOXTR, labeled COMMON area to communicate values such as longitudes, latitudes, section side lengths, geographic coordinates in DMS form, footages, the nonstandard indicator, and the closest point or corner value. Variables in the LEOXTR COMMON area are:

COMMON /LEOXTR/ LONX (4,3), LATX (4,3), FTNS, FTEW, NONSTD, CORNR

where:

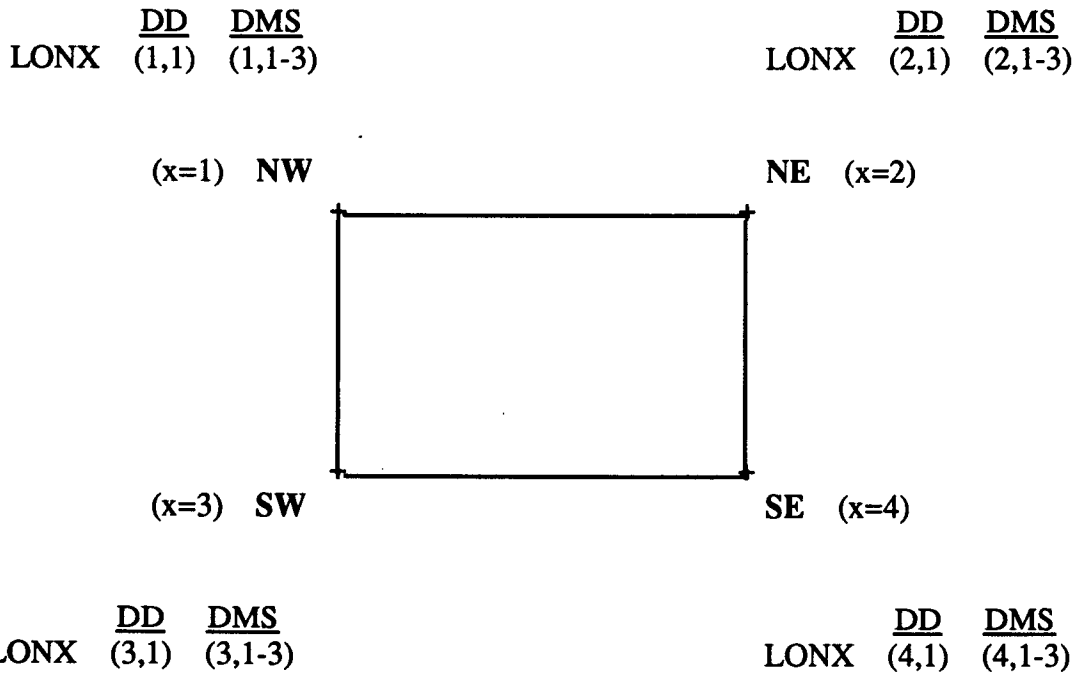
> **LONX**(4,3) - holds longitudes for geographic locations. When the option is to return one geographic location in decimal degrees, the longitude is returned in LONX (1,1), as well as in the argument PLON. When a single location is returned in DMS form, the degrees, minutes, and seconds of the longitude are returned in LONX (1,1-3). Whole degrees are

stored in LONX (1,1), whole minutes in LONX (1,2) and whole and fractional seconds in LONX (1,3). To convert a geographic location from DMS, the degrees, minutes, and seconds of longitude are placed in LONX (1,1-3). LONX(4,3) is a REAL*4 array.

When the four corners option is selected with output in decimal degrees, the longitudes of the four corners are stored in LONX (1-4,1) in the order N, E, W, S. For DMS, degrees, minutes and seconds of longitude are returned in LONX (1-4,1-3) as four groups of three real values. These groups are the four corner positions; the three values are degrees, minutes, and seconds for each. For corner x (x = 1 for NW, 2 for NE, 3 for SW or 4 for SE) degrees, minutes and seconds of longitude are stored in columns 1-3 of LONX (x,1-3).

1. STORING SECTION CORNER LONGITUDES

The longitudes (and latitudes) of section corners are passed through the LEOXTR COMMON area in the variable LONX (and LATX), with the four corners stored as illustrated.



Note: DD = Decimal Degrees, DMS = Degrees, Minutes, Seconds

When the footage option is selected with both footages set to zero, the conversion routine returns the approximate section side lengths in feet on the ground through the LEOXTR COMMON area. They are in **LONX (1-4,1)**, in the order: North, East, West, South.

The **LATX** (4,3) variable holds the latitudes corresponding to the longitudes returned in LONX (4,3), and in the same form, either decimal degrees or DMS. When LONX holds four longitudes for the four corners option, LATX holds the latitudes, in the order shown. LATX is not used when LONX returns approximate section side lengths. LATX(4,3) is a REAL*4 array.

FTNS - the variable, set by the user, that holds the North(+) (from the south to the north) or South(-) footage (from the north to the south) value in feet on the ground. FTNS is a REAL*4 (scalar) value.

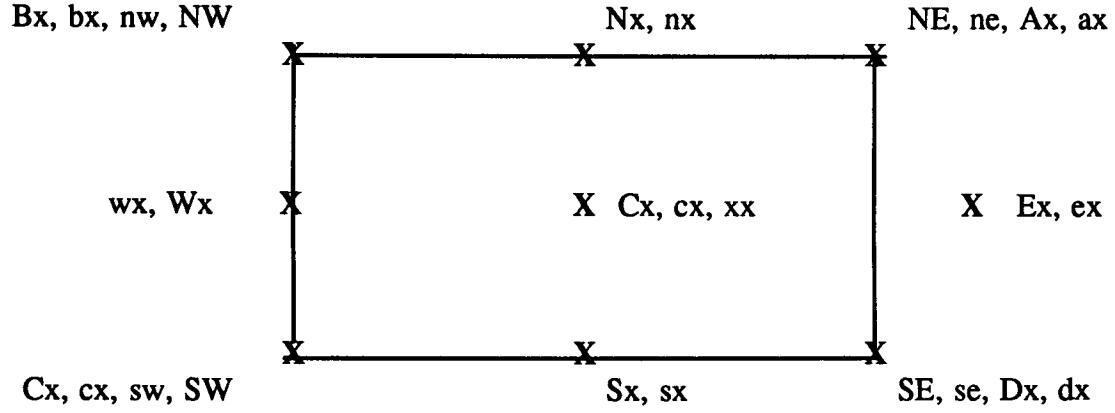
FTEW - the variable, set by the user, that holds the East(+) (from the west to the east) or West(-) footage (from the east to the west) value in feet on the ground. FTEW is also a REAL*4 (scalar) value.

NONSTD - the variable that indicates if the section involved in conversion is standard (=0) or nonstandard (=1). NONSTD is set by the program, not by the user. NONSTD is an INT*2 (scalar) value.

CORNR - a variable that indicates which of nine geometrically defined points is to be used as the location within the target area, that is closest to the conversion location - the **closest point option**. The points are the corners, side midpoints, and center of the area, identified by the appropriate quarter (corners) or half (side midpoints) area specification. When LEO2CVT creates legal output, CORNR is set to identify the closest of the nine points to the conversion location. The user selects a closest point by setting the value of CORNR in the input records to the appropriate subdivision specifier. The nine possible points thatmay be selected with CORNR, and the optional ways to select them, are shown in the illustration. CORNR is a CHAR*2 (character) variable.

2. SELECTING CLOSEST POINTS

The following illustration shows the options for selecting the nine possible closest points in the target area (corners, side midpoints, and center), the lower case "x" represents a blank.



C. CUSTOMIZED OPERATIONS

One clear advantage of modifying or rewriting the conversion program is to alter the fixed or standard formats for input and output in batch operations. These formats dictate the order of values, the content of fields, and the exact form (int/real/char) and size (field width, i.e. range of permitted values) of data in the input file and the content, order, and form of data written to the output file. The user can alter or completely redefine these formats by making changes to the conversion program, specifically, the control routine (in the LEO2PC.FOR file). Such changes create a customized conversion program that is more compatible with the local data.

Anyone with files of local data in which there are locations that are referenced either in legal or geographic coordinates, and who wants to input their data directly from that file and write output data in a similar format, with similar information attached to the locations could accomplish this by modifying the program. As the LEO II conversion program is now, this requires three steps: (1) write a program to extract the locations from the local data file and write the locations in the LEO II format for input, being careful to preserve the order of the data as in the original file (or create an index file to save order information, which is used in the third step); (2) use LEO II to convert the reference of these locations and write them in the fixed LEO II format to an output file; and (3) write a second program that reads the original, local file (or an intermediate of the information stripped from the locations in the first step) and reads the new locations, joining the two and writing the updated records to the final output file, and in the desired local format.

By making changes in the input/output (READ and WRITE) statements in the control routine of the conversion program, the conversion of a local data file can be performed in one step, rather than three, by customizing the LEO II conversion program. The customized program could read the data directly from the original, local file, convert locations from one reference system to the other, and write the original data with converted locations to the output file, in the desired, local format. For batch convrsion, only the input statement and two optional output statements, and their associated formats, need to be changed. To perform this properly, the list of variables that are read or written must be changed to reflect the content of the local files. It is likely that new variables will have to be created for the extra information (usually, other attributes) that must be read and written. Decisions about the information content and form of the input and output files are made through consultation between the programmer and the local staff.

It is possible that users also may wish to perform additional analyses, calculations, or other types of computer manipulations of their data, perhaps on the basis of the newly converted locations or additional data values. Spatially dependent analysis is not readily done using legal reference, but after the conversion, any operations may be incorporated into the conversion program. Gaining intermediate operations is another reason for modifying or rewriting the program.

- - THE END - -

APPENDIX A: STATUS VALUES

STATUS VALUE	MEANING
+2	Successful conversion in a nonstandard section (for future use)
+1	**SUCCESSFUL CONVERSION!!!**
-1	Township number out of range
-2	Range number out of range
-3	Illegal EW indicator value
-4	Section number out of range
-5	Unacceptable Subdivision #1
-6	Unacceptable Subdivision #2
-7	Unacceptable Subdivision #3
-8	Unacceptable Subdivision #4
-9	Illegal SOURCE value
-10	Invalid longitude for LEO2/Kansas
-11	Invalid latitude for LEO2/Kansas
-12	Inconsistent or invalid OPTION value
-13*	I/O Error reading LEO2BASE
-14*	Thrashing error at township level
-15*	Thrashing error at section level
-16	Illegal Footage (too large/small)
-17	Illegal DMS value in LEO2XTR
-18	Missing SUBD value for OPTION = 1, 5
-19	No usable sections in township record
-20*	Point-in-Polygon error
-21	One section/township corner missing
-22	Two section/township corners missing
-23	Three section/township corners missing
-24	Four section/township corners missing
-25	Unacceptable CORNR specification
-26	Illegal option for townships
-27*	Illegal record # in township search
-28*	Failed because of missing/unknown corners
-99*	Section requires nonstandard subdivision

* PLEASE REPORT MARKED ERRORS TO: KANSAS GEOLOGICAL SURVEY
1950 CONSTANT AVE., LAWRENCE, KS
TELE: (913) 864-3965 FAX: (913) 864-5317

APPENDIX B: REFERENCES

Collins, David R.,(1989)
The Public Land Survey System in Kansas, A Brief Historical Review: Kansas Geol. Survey, Open File Report 89-24, 5p.

Collins, David R., and Wong, J. C., (1990)
Analysis of PLSS data in Kansas: Kansas Geological Survey, maps, oral report, and detailed print-out.

Collins, David R., and Ross, C. G., (1993)
Automated conversion between geograpnic and PLS reference systems: methods and problems: Kansas Geol. Survey Open File Report 93-58, (in revision).

Hurn, Jeff, (1989)
GPS (Global Positioning System): A Guide to the Next Utility: Trimble Navigation, Sunnyvale, CA, 76p.

Good, Donald I., (1964)
Mathematical Conversion of Section, Township, and Range Notation to Cartesian Coordinates: State Geol. Survey of Kansas, Bull. 170, Pt. 30p.

Moore, John M., (1855)
"1855 Manual of Surveying Instructions": General Land Office, U.S. Printing Service, approx. 94p.

Morgan, Charles O., and McNellis, Jesse M., (1969)
FORTRAN IV Program, KANS, for the Conversion of General Land Office Locations to Latitude and Longitude Coordinates: State Geol. Survey of Kansas, Special Distribution Pub. 42, 24p.

Robinson, Arthur H., (1984)
Elements of Cartography, Fifth Ed.: John Wiley & Sons, p63.

Ross, Charles, G., (1988a)
Mapping by Computer - Introduction and GIMMAP View: Kansas Geological Survey, Open File Report 88-45a, 88p.

" , (1988b)
The GIMMAP System: Kansas Geological Survey, Open File Report 88-45b, 166p.

Ross, Charles G., (1988c)
GIMMAP Software and Technical Information: Kansas Geological Survey, Open File Report 88-45c, 253p.

" , (1988d)
Digitizing & Operations With GIMMAP: Kansas Geological Survey, Open File Report 88-45b, 166p.

" , (1989a)
PROJCT - Projection and Deprojection of Cartographic Data at the KGS: Kansas Geol. Survey, Open File Report 89-9, 8p.

" , (1989b)
LEO - Conversion Between Legal and Geographic Reference in Kansas: Kansas Geol. Survey, Open File Report 89-10, 7p.

" , (1991)
LEO II - Automated Legal and Geographic Reference System Conversion in Kansas: Kansas Geol. Survey, Open File Report 91-54, 68p, (revised 1992).

White, C. Albert, (undated, c.1982)
A History of the Rectangular Survey System: U.S. Dept. of the Interior, Bureau of Land Management, U.S. Government Printing Office, 774p.

Wilford, John Noble, (1981), The Mapmakers: Random House Inc., New York, 414p.

APPENDIX C: PROGRAM LISTINGS

NOTE: THE LEO II CONVERSION ROUTINE, LEO2CVT, IS THE CENTRAL CORE OF THE LEO II SYSTEM, AND IS LISTED IN FULL. ALL OTHER ROUTINES ARE MUCH SMALLER CAN BE EXAMINED FROM THE SOURCE FILES ON THE LEO II DISK.

```
C (MODIFIED 20 OCT 93 AND 10 JAN 94 BY CGR)
C
C*************** LEO2CVT *
C
C     CONVERSION BETWEEN LEGAL AND GEOGRAPHIC REFERENCE IN KANSAS
C
      SUBROUTINE LEO2CVT (TSHP,RANG,EW,SECT,OPT,SBD,PLON,PLAT,
     &  LUN,STAT,SORCE)
C
C ***** ***** ***** ***** DECLARATIONS **
C
      IMPLICIT NONE
C
C********** CHARACTER **
C
      CHARACTER EW*1,SBD*2(4),CORNR*2
C
C********** DOUBLE PRECISION **
C
      DOUBLE PRECISION SOUTH,NORTH,CLON,EAST,WEST,XB,YB,BASE,
     &  SCALE,CORN(9),DBLON,DBLAT,XDBL,YDBL
C
C********** INTEGER **
C
      INTEGER*2 TSHP,RANG,SECT,STAT,SORCE,LUN,TREC,I,J,SROW,
     &  SCOL,N,S,W,E,VZT(2380),VZTD(2380),RTOP,LEVEL,TROW,TCOL,
     &  SEARCH(6,6),OUTFLG,OPT,TFLAG,SFLAG(36),LEGAL,GEOGR,ORDER,
     &  NUM,OFF,NSUB,NONSTD
C
C********** LOGICAL **
C
      LOGICAL TRCE/.FALSE./
C
C********** REAL **
C
      REAL PLON,PLAT,LON(7,7),LAT(7,7),LONNE,LATNE,LONNW,LATNW,
     &  LONSW,LATSW,LONSE,LATSE,CTRLON,CTRLAT,DLON,DLAT,XLAT,XLON,
     &  LATOMN,LATOMX,LATIMN,LATIMX,LONOMN,LONOMX,LONIMN,LONIMX,
     &  WSTLON(7),DELON(7),DELAT,BASLAT,FTNS,FTEW,LONX,LATX,
     &  FTMX/5280./,LONDEG,LONMIN,LONSEC,LATDEG,LATMIN,LATSEC,
     &  XSW,YSW,XSE,YSE,XNE,YNE,XNW,YNW,RATNS,RATEW,LENGNS,LENGEW,
     &  LONINT,LATINT,LONW,LATW,LONE,LATE,LONS,LATS,LONN,LATN,
     &  LONOFF,LATOFF
```

```
C
C****** COMMON **
C
      COMMON/LEOXTR/LONX(4,3),LATX(4,3),FTNS,FTEW,NONSTD,CORNR
      COMMON/LEOMAP/SOUTH,NORTH,CLON,EAST,WEST,XB,YB,BASE,SCALE
C
C****** DATA **
C
      DATA BASLAT/40.0021/,DELAT/0.0145378/,VZT/2380*0/,VZTD/2380*0/,
     &  WSTLON/-102.19857,-102.1787,-102.14777,-102.12151,-102.09101,
     &  -102.07046,-102.0443/,
     &  DELON/.0186716,.0186243,.0185299,.0184235,.0183046,.0182329,
     &  .0182083/, CORNR/'  '/, SCALE/12.D0/
C
C *****   *****   *****   *****   *****   *****   *****
C
C                 VARIABLES:
C                 ----------
C       TSHP = TOWNSHIP NUMBER, 1-35
C       RANG = RANG NUMBER, 1-25 IF EW='E', 1-43 IF EW='W'
C       EW = RANG DIRECTION FROM CENTRAL MERIDIAN 'E' OR 'E'=EAST,
C            'W' OR 'W'=WEST
C       SECT = SECTION NUMBER, 1-36, =0 TO SPECIFY THE TOWNSHIP
C            AS THE AREA FOR CORNER/SIDE MIDPOINT SELECTION OR AS
C            THE FOUR CORNER LOCATIONS FOR THE 4-CORNER OPTION
C       OPT=SPECIFIES METHOD OF LOCATING POINT IN THE TARGET AREA
C            WITHIN THE SECTION (SEE SUBDIVISION)  WHEN SORCE=1,
C            THE OPTION IMPLIED BY THE OPT VALUE IS:
C
C            0=SELECT THE CENTER OF THE AREA DEFINED BY THE FOUR
C               CORNERS OF THE SMALLEST AREA SPECIFIED BY SBD
C               (MAY USE CORNR TO SELECT CLOSEST SIGNIFICANT POINT)
C            1=SELECT THE CORNER/SIDE MIDPOINT SPECIFIED IN SBD
C               OF A SELECTED SECTION (SECT) OR TOWNSHIP (SECT=0)
C            2=APPLY GIVEN FOOTAGES FROM IMPLIED SECTION CORNER
C               (OR RETURN SECTION SIDE LENGTHS WHEN FOOTAGES=0)
C            3=RETURN THE LOCATIONS OF THE FOUR CORNERS OF THE
C               SPECIFIED SECTION OR THE TOWNSHIP (SECT=0)
C            4=SELECT THE CENTER (AS IN 0), BUT IN DEGREES,
C               MINUTES,AND SECONDS OF LON/LAT,THE DMS OPTION
C               (MAY USE CORNR TO SELECT CLOSEST SIGNIFICANT POINT)
C            5=(AS IN 1) SELECT THE SPECIFIED CORNER OR SIDE
C               MIDPOINT OF A SECTION, BUT RETURN IN DMS FORM
C            6=APPLY FOOTAGES FROM THE IMPLIED SECTION OR TOWNSHIP
C               CORNER, BUT RETURN IN DMS FORM IN LONX (1,1-3)
C               AND LATX (1,1-3)
C            7=RETURN THE SECTION OR TOWNSHIP FOUR CORNERS,
C               BUT IN THE DMS FORM
C
C       NOTE: FOR OPT=0-2, THE GEOGRAPHIC LOCATION IS RETURNED VIA PLON
C         AND PLAT WITH A POSITIVE LONGITUDE.  ALL OTHER OPTIONS USE
C         LONX AND LATX TO RETURN 1 OR 4 VALUES, (ALL IN DMS EXCEPT
C         THE BASIC 4-CORNERS OPTION=3), WITH POSITIVE LONGITUDE(S)
C
C       NOTE: WHEN SORCE=2,THE OPTION IMPLIED BY THE OPT VALUE IS
C
C            0=INPUT IS IN DECIMAL DEGREES (IN PLON < 0, PLAT)
C            1=INPUT IS IN THE DMS FORM (IN LONX > 0, LATX)
C
```

```
C     SBD=SPECIFY 1-4 SUBDIVISIONS OF THE SECTION,EACH APPLIED TO
C     THE RESULT OF THE PREVIOUS ONES.  SUBDIVISIONS ARE DEFINED AS
C     SELECTION OF A HALF OR QUARTER OF THE AREA BEING DIVIDED,AND
C     ARE SPECIFIED BY LETTERS REPRESENTING POINTS OF THE COMPASS:
C
C     1/4 AREA....   NE OR AB....NORTHEAST  (B=BLANK)
C                    NW OR BB....NORTHWEST
C                    SW OR CB....SOUTHWEST
C                    SE OR DB....SOUTHEAST
C                      BB.....UNUSED
C     1/2 AREA..... NB OR N2.....NORTH
C                    SB OR S2.....SOUTH
C                    EB OR E2.....EAST
C                    WB OR W2.....WEST
C                      BB.....UNUSED
C
C     SBD(1)=THE LARGEST (1/N SECTION) SUBDIVISION,AND THE ONE
C            THAT IS APPLIED FIRST (TO THE WHOLE SECTION).
C     SBD(2)=THE SECOND LARGEST (1/N-1/N SECTION) SUBDIVISION.
C     SBD(3)=THE SECOND SMALLEST (1/N-1/N-1/N SECTION) SUBDIVISION
C     SBD(4)=THE SMALLEST (BY AREA 1/N-1/N-1/N-1/N SECTION) AND THE
C            LAST SUBDIVISION TO BE APPLIED (TO THE RESULT OF THE
C            PRECEEDING THREE SUBDIVISIONS IN SBD(1-3))
C            WHERE 1/N REPRESENTS EITHER 1/2 OR 1/4
C              (WHERE EACH OF THE 1/N IS EITHER 1/2 OR 1/4)
C
C     PLON = LONGITUDE OF POINT FOR OR FROM CONVERSION (POSITIVE)
C     PLAT = LATITUDE OF POINT
C     LUN = UNIT NUMBER OF (PREVIOUSLY OPENED) LEOBASE
C     STAT = CODE INDICATING SUCCESS OR FAILURE OF THE REQUESTED
C       OPERATION.  THE CODES ARE:
C
C            1=SUCCESS
C           -1=TOWNSHIP NUMBER OUT OF RANG
C           -2=RANG NUMBER OUT OF RANG
C           -3=RANG DIRECTION (EW) NOT 'E' OR 'W'
C           -4=SECTION NUMBER OUT OF RANG
C           -5=ILLEGAL/UNRECOGNIZED SUBDIVISION SPEC  #1
C           -6=  "     "        "    "   #2
C           -7=  "     "        "    "   #3
C           -8=  "     "        "    "   #4
C           -9=ILLEGAL CONVERSION DIRECTION (SORCE) VALUE
C          -10=INVALID LONGITUDE FOR LEO/KGS
C          -11=INVALID LATITUDE FOR LEO/KGS
C          -12=INVALID VALUE FOR OPT
C          -13=DATABASE (LEOBASE) I/O ERROR
C          -14=THRASHING ERROR ON THE TOWNSHIP LEVEL
C          -15=THRASHING ERROR ON THE SECTION LEVEL
C          -16=ILLEGAL FOOTAGE (TOO LARGE OR TOO SMALL)
C          -17=ERROR IN DEGREES,MINUTES,OR SECONDS VALUE
C          -18=MISSING SBD VALUE FOR OPT=1 OR 5
C          -19=TOWNSHIP RECORD HAS NO USABLE SECTIONS
C          -20=POINT-IN-POLYGON ERROR
C          -21=ONE SECTION OR TOWNSHIP CORNER IS MISSING
C          -22=TWO SECTION OR TOWNSHIP CORNERS ARE MISSING
C          -23=THREE SECTION OR TOWNSHIP CORNERS ARE MISSING
C          -24=ALL FOUR SECTION OR TOWNSHIP CORNERS ARE MISSING
C          -25=ILLEGAL OR UNRECOGNIZED CORNER SPECIFICATION
C          -26=OPTION IS NOT ALLOWED FOR TOWNSHIPS
```

```
C          -27=ILLEGAL RECORD # IN TOWNSHIP SEARCH
C          -28=FAILED SEARCH WITHIN INCOMPLETE TOWNSHIP
C          -99=SPECIAL SECTION REQUIRES NON-STANDARD SUBDIVISION
C
C     SORCE = SORCE, INDICATES DIRECTION OF THE CONVERSION:
C             1=LEGAL, CVT FROM LEGAL TO GEOGRAPHIC
C             2=GEOGRAPHIC, CVT FROM GEOGRAPHIC TO LEGAL
C
C     LONX = HAS LONGITUDES IN DEGREES, MINUTES, SECONDS FOR THE
C     LATX    DMS OPTION, IF SORCE=2, THE GEOGRAPHIC LOCATION
C       IS STORED IN LONX AND  LATX WITH DEGREES OF LONGITUDE IN
C       LONX(1,1) AND MINUTES AND SECONDS IN LONX(1,2-3). THE
C       CORRESPONDING DEGREES,MINUTES AND SECONDS OF LATITUDE ARE
C       STORED IN LATX(1,1-3). WHEN SORCE=1 (LEGAL),THE
C       CONVERTED OUTPUT LOCATION IS STORED IN THE SAME MANNER,AS
C       WELL AS IN PLON AND PLAT. FOR THE 4 CORNERS OPTION,THE FOUR
C       OUTPUT CORNER LOCATIONS ARE STORED SIMILARLY WITH THE NW
C       CORNER IN LON/LATX(1,1-3),THE NE CORNER IN LON/LATX(2,*)
C       THE SW IN (3,*) AND THE SE IN (4,*).
C     FTNS = FOOTAGE APPLIED TO THE NORTH IF > 0, SOUTH (IF < 0)
C     FTEW = FOOTAGE APPLIED TO THE EAST IF > 0, WEST IF < 0
C       NOTE: FOOTAGES ARE APPLIED FROM THE SECTION OR TOWNSHIP
C         CORNER IMPLIED BY THE SIGNS OF THE FOOTAGE VALUES
C       NOTE: WHEN BOTH FOOTAGES ARE 0,THE LENGTHS OF THE SECTION
C         SIDES IN FEET ON THE GROUND ARE RETURNED IN LONX (1-4,1)
C     NONSTD = 1, IF THE SECTION INVOLVED IN EITHER CONVERSION IS
C       IN SOME WAY IRREGULAR AND HAS BEEN MARKED TO INDICATE THAT
C       SUBDIVISION OF THE SECTION SHOULD BE PERFORMED IN A NON-STD
C       MANNER (I.E. OTHER THAN THE STANDARD FOUR EQUAL PARTS SUBD.)
C      = 0,IF THE SECTION IS SUBDIVIDED IN THE STANDARD FASHION
C     CORNR = IF SOURC=1 (LEGAL), THIS VALUE, IN THE SAME FORM AS
C       THE SBD VALUES, SPECIFIES ONE OF NINE POINTS IN THE TARGET
C       AREA-THE AREA DEFINED BY THE LEGAL DEFINITION,INCLUDING THE
C       (UP TO 4) SUBDIVISIONS IN SBD. THE NINE POINTS ARE THE FOUR
C       CORNERS,THE SIDE MIDPOINTS (S,N...) AND THE DEFAULT CENTER.
C       WHEN SORCE=2 (GEOGRAPHIC),THIS OUTPUT VALUE IS SET TO
C       INDICATE THE CLOSEST OF THE NINE POINTS WITHIN THE FOURTH
C       SUBDIVISION,TO THE POINT DESCRIBED IN PLON,PLAT
C     TFLAG = TOWNSHIP FLAG, = 0 IF ALL 36 SECTIONS HAVE 4 DEFINED
C       CORNERS, = -1 IF NO SECTIONS HAVE 4 DEFINED CORNERS, OR = N,
C       0< N <36, N = NUMBER OF SECTIONS WITH 4 DEFINED CORNERS
C     SFLAG = SECTION FLAG, = 0 FOR SECTIONS WITH 4 DEFINED CORNERS,
C       OR = -N, WHERE N = NUMBER OF MISSING CORNERS.
C     XC,YC = PROJECTED LOCATIONS OF THE 49 CORNERS OF A TOWNSHIP
C       IN THE SAME ORDER AS THE LONGITUDES AND LATITUDES IN LON,LAT
C   (T/S)ROW = ROW NUMBER OF THE TOWNSHIP OR SECTION WITHIN A TOWNSHIP
C       BASED ON AN ARTIFICIAL GRID. FOR TOWNSHIPS,THE GRID IS 35
C       ROWS NUMBERED FROM THE NORTH,AND 68 COLUMNS NUMBERED FROM
C       THE WEST. FOR SECTIONS,THE GRID IS 7 BY 7 REPRESENTING THE
C       49 CORNER POINTS WITH ROW 1 AT THE NORTH,COLUMN 1 AT THE WEST
C   (T/S)COL = ASSOCIATED COLUMN NUMBER OF THE TOWNSHIP OR SECTION
C     LONNE = LONGITUDE OF NE CORNER OF SECTION OR SUBDIVISION
C     LATNE = LATITUDE OF SAME
C     LONSE = LONGITUDE OF SE CORNER OF SAME
C     LATSE = LATITUDE OF SAME
C     LONSW = LONGITUDE OF SW CORNER
C     LATSW = LATITUDE OF SAME
C     LONNW = LONGITUDE OF NW CORNER
C     LATNW = LATITUDE OF SAME
```

```
C     OUTFLG = CODE LIKE A CLIPPING CODE INDICATES IF A POINT IS IN
C       OR OUT AND WHERE IF OUT OF A RECTANGLE
C     LONOFF = OFFSET FROM SECTION CORNER, IN LONGITUDE, CALCULATED
C       AS THE RATIO OF THE NS FOOTAGE TO THE AVERAGE LENGTH OF THE
C       NORTH AND SOUTH SIDES,MULTIPLIED BY THE CHANGE IN LONGITUDE
C       FROM THE EAST TO THE WEST ALONG THOSE SIDES BUT SCALED
C     LATOFF = THE LATITUDE EQUIVALENT OF LONOFF
C     LONOMN = MINIMUM LONGITUDE OF OUTSIDE RECTANGLE
C     LATOMN =  "  LATITUDE OF SAME
C     LONOMX = MAXIMUM LONGITUDE OF SAME
C     LATOMX =  "  LATITUDE OF SAME
C     LONIMN = MINIMUM LONGITUDE OF INSIDE RECTANGLE
C     LATIMN =  "  LATITUDE OF SAME
C     LONIMX = MAXIMUM LONGITUDE OF SAME
C     LATIMX =  "  LATITUDE OF SAME
C     DLON = LENGTH OF SIDE OF TRIANGLE SOLVED TO FIND XLON,THE
C      LONGITUDE OF AN INTERSECTION POINT
C     DLAT = SAME BUT IN LATITUDE
C     XLON = LONGITUDE OF THE INTERSECTION POINT OF A VERTICAL RAY
C      PASSING THROUGH THE POINT WITH A SIDE OF THE QUADRILATERAL
C     XLAT = LATITUDE OF RAY INTERSECTION THROUGH POINT WITH SIDE
C
C******* CHECK THE INPUT VALUES FOR ERRORS FOR BOTH CONVERSION TYPES
C
   LEGAL=1
   GEOGR=2
   NONSTD=0
   STAT=0
C*************************************************************************
C* * * * * * * * * * * * * * * * * * * * * * CHECK LEGAL INPUT VALUES
C*************************************************************************
C
   IF (SORCE.EQ.LEGAL) THEN
C
C****  INITIALIZE OUTPUT VARIABLES TO SAFE VALUES AND
C*     CHECK FOR ERRORS IN LEGAL INPUT PARAMETERS
C
      PLON=0.
      PLAT=0.
      LONX(1,1)=0.
      LATX(1,1)=0.
      IF (TSHP.LT.1 .OR. TSHP.GT.35) THEN
        STAT=-1
        RETURN
      ENDIF
      IF (EW.EQ.'E' .OR. EW.EQ.'E') THEN
        IF (RANG.LT.1 .OR. RANG.GT.25) THEN
        STAT=-2
        RETURN
        ENDIF
        EW='E'
      ELSEIF (EW.EQ.'W' .OR. EW.EQ.'w') THEN
        IF (RANG.LT.1 .OR. RANG.GT.43) THEN
        STAT=-2
        RETURN
        ENDIF
        EW='W'
      ELSE
        STAT=-3
```

```
        RETURN
      ENDIF
      IF (OPT.LT.0 .OR. OPT.GT.7) THEN
        STAT=-12
        RETURN
      ENDIF
      IF (SECT.LT.0 .OR. SECT.GT.36) THEN
        STAT=-4
        RETURN
      ENDIF
C
C*  CHECK FOR ILLEGAL FOOTAGE VALUES.
C
      IF (OPT.EQ.2 .OR. OPT.EQ.6) THEN
        IF (ABS(FTNS).GT.FTMX .OR. ABS(FTEW).GT.FTMX) THEN
        STAT=-16
        RETURN
        ENDIF
C
C*  FOR CORNER/SIDE OPTION CHECK FOR SPEC IN SBD(1) OR CORNR
C*  IF CORNR IS BLANK BUT SBD IS NOT, COPY SBD TO CORNR
C
      ELSEIF (OPT.EQ.1 .OR. OPT.EQ.5) THEN
        IF (CORNR.EQ.'  '.AND. SBD(1).NE.'  ') CORNR = SBD(1)
C
C*******************  CHECK SUBDIVISION SPECS (OPT = 0/4)
C
      ELSEIF (OPT.EQ.0 .OR. OPT.EQ.4) THEN
C
C*  FOR SUBDIVISION OPTIONS, CHECK SBD
C*  VALUES AND SET ORDER TO GUIDE THE SUBDIVISION PROCESS
C
C
C*********************  CHECK AND CODE SUBDIVISIONS
C
C
C*  NUMBER OF SUBDIVISIONS USED INITIALLY ASSSUMED TO BE 4, PROCESS
C*  CONTROL IS INITIALLY ZERO, AND UNUSUAL CENTER SPECS FOR CORNR ARE
C*  RESET TO BLANK.  PROCESS CONTROL VIA ORDER IS CREATED BY ADDING
C*  ONE DIGIT IN EACH OF 0 TO 4 DECIMAL POSITIONS IN ORDER, WITH EACH
C*  DIGIT UNIQUELY REPRESENTING ONE OF EIGHT POSSIBLE SUBDIVISION SPECS
C
        NSUB=4
        ORDER=0
        IF (CORNR.EQ.'C ' .OR. CORNR.EQ.'c ') CORNR = '  '
        IF (CORNR.EQ.'CR' .OR. CORNR.EQ.'cr') CORNR = '  '
        DO 50 I=4,1,-1
        IF(SBD(I)(1:1).EQ.' ') THEN
          SBD(I)=SBD(I)(2:2) // SBD(I)(1:1)
        ENDIF
        IF (SBD(I).EQ.'D ' .OR. SBD(I).EQ.'SE') THEN
          ORDER=10*ORDER+1
        ELSEIF (SBD(I).EQ.'D ' .OR. SBD(I).EQ.'SE') THEN
          SBD(I) = 'SE'
          ORDER=10*ORDER+1
        ELSEIF (SBD(I).EQ.'A ' .OR. SBD(I).EQ.'NE') THEN
          ORDER=10*ORDER+2
        ELSEIF (SBD(I).EQ.'A ' .OR. SBD(I).EQ.'NE') THEN
          SBD(I) = 'NE'
```

```
          ORDER=10*ORDER+2
        ELSEIF (SBD(I).EQ.'B ' .OR. SBD(I).EQ.'NW') THEN
          ORDER=10*ORDER+3
        ELSEIF (SBD(I).EQ.'B ' .OR. SBD(I).EQ.'NW') THEN
          SBD(I) = 'NW'
          ORDER=10*ORDER+3
        ELSEIF (SBD(I).EQ.'C ' .OR. SBD(I).EQ.'SW') THEN
          ORDER=10*ORDER+4
        ELSEIF (SBD(I).EQ.'C ' .OR. SBD(I).EQ.'SW') THEN
          SBD(I) = 'SW'
          ORDER=10*ORDER+4
        ELSEIF (SBD(I).EQ.'N ' .OR. SBD(I).EQ.'N ') THEN
          SBD(I) = 'N '
          ORDER = 10 * ORDER + 5
        ELSEIF (SBD(I).EQ.'W ' .OR. SBD(I).EQ.'W ') THEN
          SBD(I) = 'W'
          ORDER=10*ORDER+6
        ELSEIF (SBD(I).EQ.'S ' .OR. SBD(I).EQ.'S ') THEN
          SBD(I) = 'S'
          ORDER=10*ORDER+7
        ELSEIF (SBD(I).EQ.'E ' .OR. SBD(I).EQ.'E ') THEN
          SBD(I) = 'E'
          ORDER=10*ORDER+8
        ELSEIF (SBD(I).EQ.' ' .AND. ORDER.EQ.0) THEN
          NSUB=I-1
          IF (I.LT.1) NSUB=1
        ELSE
          STAT=-4-I
          RETURN
        ENDIF
50        CONTINUE
      ENDIF
C
C ********************* CHECK GEOGRAPHIC INPUT VALUES
C*
C** NOW CHECK THE ARGUMENTS FOR GEOGRAPHIC TO LEGAL (SORCE=2)
C*
C
    ELSEIF (SORCE.EQ.GEOGR) THEN
      TSHP=0
      RANG=0
      SECT=0
      EW=' '
      SBD(1)=' '
      SBD(2)=' '
      SBD(3)=' '
      SBD(4)=' '
      CORNR=' '
C
C* IF OPT=1,CONVERT DMS VALUES IN LONX(1) AND LATX(1)
C*  TO PLON,PLAT AFTER CHECKING THE DMS VALUES FOR ERRORS
C
      IF (OPT.EQ.1) THEN
        LONDEG=ABS(LONX(1,1))
        LONMIN=ABS(LONX(1,2))
        LONSEC=ABS(LONX(1,3))
        LATDEG=LATX(1,1)
        LATMIN=LATX(1,2)
        LATSEC=LATX(1,3)
```

```
        IF (LONDEG.LT.94. .OR. LONDEG.GT.102.) THEN
        STAT=-17
        RETURN
        ELSEIF (LONMIN.GT.59. .OR. LONSEC.GT.59.) THEN
        STAT=-17
        RETURN
        ELSEIF (LONMIN.LT.0. .OR. LONSEC.LT.0.) THEN
        STAT=-17
        RETURN
        ELSEIF (LATDEG.LT.36. .OR. LATDEG.GT.40.) THEN
        STAT=-17
        RETURN
        ELSEIF (LATMIN.GT.59. .OR. LATMIN.LT.0.) THEN
        STAT=-17
        RETURN
        ELSEIF (LATSEC.GT.59. .OR. LATSEC.LT.0.) THEN
        STAT=-17
        RETURN
        ENDIF
        PLON=LONDEG+LONMIN/60.+LONSEC/3600.
        PLAT=LATDEG+LATMIN/60.+LATSEC/3600.
      ENDIF
C
C*  MAKE SURE THAT THE INPUT LONGITUDES ARE SET NEGATIVE FOR
C*  INTERNAL PROCESSING (OUTPUT VALUES WILL BE POSITIVE),AND
C*  ALSO CHECK THE LONG/LAT EXTREMES FOR KANSAS
C
      IF (PLON.GT.0.) PLON=-1.*PLON
      IF (PLON.LT.-102.05 .OR. PLON.GT.-94.55) THEN
        STAT=-10
        RETURN
      ENDIF
      IF (PLAT.LT.36.95 .OR. PLAT.GT.40.05) THEN
        STAT=-11
        RETURN
      ENDIF
    ELSE
      STAT=-9
      RETURN
    ENDIF
C
C*
C     ** FIRST HANDLE THE OPTION SORCE=1. SORCE INFORMATION
C     ** IS LEGAL (TOWNSHIP/RANG/SECTION ETC.) AND THE DESIRED
C     ** OUTPUT IS GEOGRAPHIC (LONGITUDE,LATITUDE).
C*
C
C*************************************************************************
C * * * * * * * * * * * * * * * * * * * LEGAL TO GEOGRAPHIC CONVERSION
C*************************************************************************
C
C
C * * * * * * * * * FIND CORNER LOCATIONS OF SECTION/TSHP
C
C*  CALCULATE THE RECORD NUMBER AND READ THE TOWNSHIP RECORD
C
    IF (SORCE.EQ.LEGAL) THEN
      TREC=(TSHP-1)*68
      IF (EW.EQ.'E') THEN
```

```
        TREC=TREC+43+RANG
      ELSE
        TREC=TREC+44-RANG
      ENDIF
      READ(LUN,REC=TREC,ERR=1999)
     &    ((LON(I,J),LAT(I,J),J=1,7),I=1,7), TFLAG, SFLAG
C
C*  CHECK FOR EMPTY TOWNSHIP
C
      IF (TFLAG.EQ.-1) THEN
        STAT=-19
        RETURN
      ENDIF
      IF (SECT.NE.0) THEN
C
C*  CHECK FOR MISSING SECTION CORNERS
C
        IF (SFLAG(SECT).LT.0) THEN
        STAT=SFLAG(SECT)-20
        RETURN
        ENDIF
C
C*  GET THE ROW AND COLUMN OF THE NW CORNER OF DESIRED SECTION
C
        SROW=(SECT-1)/6+1
        SCOL=MOD (INT (SECT),6)
        IF (SCOL.EQ.0) SCOL=6
        IF (MOD (INT (SROW),2).EQ.1) SCOL=7-SCOL
      ELSE
C
C*  CHECK FOR MISSING TOWNSHIP CORNERS (SECT=0)
C
        STAT=0
        DO 75 I=1,7,6
        DO 75 J=1,7,6
          IF (LAT(I,J).LT.0.) STAT=STAT-1
75        CONTINUE
        IF (STAT.LT.0) THEN
        STAT=STAT-20
        RETURN
        ENDIF
C
C*  CHECK FOR FOOTAGE OR SUBDIVISION OPTIONS WHICH ARE NOT
C*  ALLOWED WHEN THE TOWNSHIP HAS BEEN SELECTED
C
        IF (MOD (INT (OPT),2).EQ.0) THEN
        STAT=-26
        RETURN
        ENDIF
C
C*  SET SROW=SCOL=1 FOR THE TOWNSHIP (NW CORNER)
C
        SROW=1
        SCOL=1
      ENDIF
C
C**************************************
C ********************* FOUR CORNERS OPTION (OPT = 3/7)
C**************************************
```

```
C*
C** DO THE FOUR CORNERS OPTION FIRST (OPT=3,7)
C*
C
      IF (OPT.EQ.3 .OR. OPT.EQ.7) THEN
        DO 100 I=1,3,2
        LONX(I,1)=LON(SROW,SCOL)
        LATX(I,1)=LAT(SROW,SCOL)
        IF (SECT.NE.0) THEN
          SCOL=SCOL+1
        ELSE
          SCOL=7
        ENDIF
        LONX(I+1,1)=LON(SROW,SCOL)
        LATX(I+1,1)=LAT(SROW,SCOL)
        IF (SECT.NE.0) THEN
          SROW=SROW+1
          SCOL=SCOL-1
        ELSE
          SROW=7
          SCOL=1
        ENDIF
100         CONTINUE
C
C*
C* * * * * * * * * * * * * * * * CONVERT FOUR CORNERS TO DMS  (OPT = 7)
C*
C
C**** CONVERT THE FOUR CORNERS TO DMS IF REQUESTED (OPT=7)
C
        IF (OPT.EQ.7) THEN
        DO 200 I=1,4
C
C* CONVERT THE 4 LONGITUDES IN DEGREES IN LONX(1-4,1) INTO
C* DEGREES (1-4,1),MINUTES (1-4,2),AND SECONDS (1-4,3).
C* START BY SETTING THE LONGITUDES TO POSITIVE VALUES SINCE
C* ALL DEGREE-MINUTE-SECOND VALUES MUST BE POSITIVE.
C
          LONX(I,1)=ABS(LONX(I,1))
          LONX(I,2)=60.*(LONX(I,1)-INT (LONX(I,1)))
          LONX(I,1)=INT (LONX(I,1))
          LONX(I,3)=60.*(LONX(I,2)-INT(LONX(I,2)))
          LONX(I,2)=INT (LONX(I,2))
          IF (LONX(I,3).GE.60.) THEN
            LONX(I,3)=LONX(I,3)-60.
            LONX(I,2)=LONX(I,2)+1.
          ENDIF
          IF (LONX(I,2).GE.60.) THEN
            LONX(I,2)=LONX(I,2)-60.
            LONX(I,1)=LONX(I,1)+1.
          ENDIF
C
C* CONVERT THE 4 LATITUDES IN DEGREES IN LATX(1-4,1) INTO
C* DEGREES (1-4,1),MINUTES (1-4,2),AND SECONDS (1-4,3).
C
          LATX(I,2)=60.*(LATX(I,1)-INT(LATX(I,1)))
          LATX(I,1)=INT (LATX(I,1))
          LATX(I,3)=60.*(LATX(I,2)-INT(LATX(I,2)))
          LATX(I,2)=INT (LATX(I,2))
```

```
          IF (LATX(I,3).GE.60.) THEN
            LATX(I,3)=LATX(I,3)-60.
            LATX(I,2)=LATX(I,2)+1.
          ENDIF
          IF (LATX(I,2).GE.60.) THEN
            LATX(I,2)=LATX(I,2)-60.
            LATX(I,1)=LATX(I,1)+1.
          ENDIF
200          CONTINUE
        ENDIF
C
C*  GO SET STAT VALUE AND EXIT
C
        GOTO 575
      ENDIF
C
C*******************************
C * * * * * * * * * * * * * * * EXTRACT SECTION/TSHP CORNER LOCATIONS
C* * * * * * * * * * * * * * * FOR ALL OTHER OPTIONS (0/4, 1/5, 2/6)
C*******************************
C
C******* GET THE CORNER LOCATIONS FOR ALL OPTIONS EXCEPT 4 CORNERS
C
      OFF=1
      IF (SECT.EQ.0) OFF=6
      LONNW=LON(SROW,SCOL)
      LATNW=LAT(SROW,SCOL)
      LONSW=LON(SROW+OFF,SCOL)
      LATSW=LAT(SROW+OFF,SCOL)
      LONSE=LON(SROW+OFF,SCOL+OFF)
      LATSE=LAT(SROW+OFF,SCOL+OFF)
      LONNE=LON(SROW,SCOL+OFF)
      LATNE=LAT(SROW,SCOL+OFF)
C
C*
C * * * * * * * * * * * * * * * FOR CORNER/SIDE MIDPT OPTIONS (OPT=1,5)
C* * * * * * * * * * * * * * * FIND AREA CENTER AND GO TO APPLY CORNR
C*
C

      IF (OPT.EQ.1 .OR. OPT.EQ.5) THEN
        CTRLON=(LONNW+LONSW+LONSE+LONNE)/4.
        CTRLAT=(LATNW+LATSW+LATSE+LATNE)/4.
C
C*  GO TO APPLY CORNR, OPTIONAL DMS CONVERSION, AND TO SET STAT
C
        GOTO 510
      ENDIF
C
C***********************************************
C * * * * * * * * * * * * * * * * * * * * * * * FOOTAGES (OPT = 2/6)
C***********************************************
C
C*
C*  DO FOOTAGES FOR LEGAL TO GEOGRAPHIC CONVERSIONS HERE
C*
C
      IF (OPT.EQ.2 .OR. OPT.EQ.6) THEN
C
```

```
C* FIRST,SET THE PROJECTION AREA TO BE THE MINIMUM
C* BOUNDING RECTANGLE ABOUT THE SECTION CORNERS AND
C* INITIALIZE THE PROJECTION SOFTWARE WITH A FIXED MP
C* PROJECTION AT SCALE=12 TO PRODUCE UNITS EQUAL TO
C* FEET ON THE EARTH'S SURFACE.
C
C
C* * * * * * * * * * * * * * * * * PROJECT CORNERS TO FIND SIDE LENGTHS
C
C
      SOUTH=AMIN1 (LATSW,LATSE)
      NORTH=AMAX1 (LATNW,LATNE)
      WEST=AMIN1 (LONSW,LONNW)
      EAST=AMAX1 (LONSE,LONNE)
C
C* INITIALIZE THE (MP) PROJECTION
C
      CALL LEO2SETP (CORN)
C
C* PROJECT THE FOUR SECTION CORNERS (FOOTAGES ARE APPLIED ONLY
C* FROM THE SECTION CORNERS)
C
      DBLON=DBLE (LONSW)
      DBLAT=DBLE (LATSW)
      CALL LEO2PRO (DBLON,DBLAT,XDBL,YDBL)
      XSW=SNGL (XDBL)
      YSW=SNGL (YDBL)
      DBLON=DBLE (LONSE)
      DBLAT=DBLE (LATSE)
      CALL LEO2PRO (DBLON,DBLAT,XDBL,YDBL)
      XSE=SNGL (XDBL)
      YSE=SNGL (YDBL)
      DBLON=DBLE (LONNE)
      DBLAT=DBLE (LATNE)
      CALL LEO2PRO (DBLON,DBLAT,XDBL,YDBL)
      XNE=SNGL (XDBL)
      YNE=SNGL (YDBL)
      DBLON=DBLE (LONNW)
      DBLAT=DBLE (LATNW)
      CALL LEO2PRO (DBLON,DBLAT,XDBL,YDBL)
      XNW=SNGL (XDBL)
      YNW=SNGL (YDBL)
C
C******* IF FOOTAGES ARE (APPROX.) EQUAL TO ZERO,THE CALL IS TO
C*      GET THE LENGTHS OF ALL FOUR SIDES OF THE SECTION,MEASURED
C*      IN FEET ON THE GROUND (PROJECTED UNITS WHEN DONE AS IT IS
C*      FOR THE FOOTAGE APPLICATION). THESE ARE CALCULATED AND
C*      STORED IN LONX(1-4,1) IN THE ORDER N,E,W,AND S. THEN
C******* GO TO SET THE VALUE OF STAT AND NONSTD AND EXIT.
C
      IF (ABS(FTNS).LT.1. .AND. ABS(FTEW).LT.1.) THEN
      LONX(1,1)=SQRT((XNE-XNW)**2+(YNE-YNW)**2)
      LONX(2,1)=SQRT((XNE-XSE) ** 2+(YNE-YSE) ** 2 )
      LONX(3,1)=SQRT((XNW-XSW) ** 2+(YNW-YSW) ** 2 )
      LONX(4,1)=SQRT((XSE-XSW) ** 2+(YSE-YSW) ** 2 )
      GOTO 575
      ENDIF
C
C****** CALCULATE THE NS AND EW RATIOS OF FOOTAGE TO LENGTH AFTER
```

```
C*      CALCULATING THE APPROPRIATE SIDE LENGTHS WHICH ARE
C*      DETERMINED BY THE FOOTAGE DIRECTIONS THAT ARE IMPLIED
C*      BY THE SIGNS OF THE FOOTAGES (+=N,E AND -=S,W)
C****** WHICH IN TURN DETERMINES THE CORNER FOR APPLICATION
C
C
C* * * * * * * * * * * * * * * * APPLY SIDE LENGTH TO FOOTAGE RATIOS TO
C * * * * * * * * * * * * * * * LONG/LAT CHANGES TO FIND POINT LOCATION
C
C
C* LENGNS AND LENGEW ARE THE AVERAGED SIDE LENGTHS OF THE
C* SECTION. DIVIDE THE FOOTAGES BY THEM TO GET THE RATIOS.
C* (MAKE SURE THE FOOTAGES DON'T EXCEED THE LENGTHS)
C* (SIDE LENGTHS ARE MEASURED AS CHANGES IN X OR Y ONLY)
C
      LENGNS=((YNE-YSE)+(YNW-YSW))/2.
      IF (ABS(FTNS).GT.LENGNS) THEN
      STAT=-16
      RETURN
      ENDIF
      LENGEW=((XNE-XNW)+(XSE-XSW))/2.
      IF (ABS(FTEW).GT.LENGEW) THEN
      STAT=-16
      RETURN
      ENDIF
      RATNS=ABS(FTNS)/LENGNS
      RATEW=ABS(FTEW)/LENGEW
C
C* RATIOS ARE APPLIED TO SIDE LENGTHS TO FIND OFFSETS FROM THE
C* CORNER OF ORIGIN. THE LENGTH OF SIDE USED IS INDIRECTLY
C* PROPORTIONATE TO THE DISTANCE IT IS TO BE APPLIED AS MEASURED
C* FROM THE OTHER PAIR OF SIDES. THAT IS, IF FOOTAGES ARE TO BE
C* APPLIED FROM THE NORTH AND THE WEST THEN THE EAST-WEST RATIO
C* IS THE WEST FOOTAGE DIVIDED BY THE AVERAGE CHANGE IN X FROM
C* EAST TO WEST FOR THE NORTH SIDE AND THE SOUTH SIDE. THIS RATIO
C* IS APPLIED TO THE LENGTH OF THE NORTH AND SOUTH SIDES, BUT IS
C* WEIGHTED ACCORDING TO DISTANCE FROM EACH SIDE TO THE POINT AT
C* WHICH THE FOOTAGE IS TO BE APPLIED.
C
C* OFFSETS IN LONGITUDE AND LATITUDE ARE CALCULATED AS EXPLAINED
C* ABOVE (THOUGH NOT WELL)
C
      LONOFF=RATNS*(LONNE-LONNW)+(1.-RATNS)*(LONSE-LONSW)
      LONOFF=RATEW*LONOFF
      LATOFF=RATEW*(LATNW-LATSW)+(1.-RATEW)*(LATNE-LATSE)
      LATOFF=RATNS*LATOFF
C
C* THE LONG/LAT OFFSETS ARE CALCULATED SO THE RATIO OF THE
C* OFFSETS TO THE CHANGES IN LONGITUDE AND LATITUDE ACROSS THE
C* SECTION IS PARALLEL TO THE RATIOS CALCULATED ABOVE (RATNS
C* AND RATEW) USING THE PROJECTED (X,Y) COORDINATES.
C
C
C* APPLY OFFSETS FROM THE APPROPRIATE CORNER
C
      IF (FTNS.GT.0.) THEN
      IF (FTEW.GT.0.) THEN
        PLON=LONSW+LONOFF
        PLAT=LATSW+LATOFF
```

```
        ELSE
          PLON=LONSE-LONOFF
          PLAT=LATSE+LATOFF
        ENDIF
        ELSE
        IF (FTEW.GT.0.) THEN
          PLON=LONNW+LONOFF
          PLAT=LATNW-LATOFF
        ELSE
          PLON=LONNE-LONOFF
          PLAT=LATNE-LATOFF
        ENDIF
        ENDIF
C
C*  GO TO OPTIONAL DMS CONVERSION AND TO SET STAT
C
        GOTO 550
      ENDIF
C
C
C*****************************
C * * * * * * * * * * * * * * * SECTION & SUBAREA SUBDIVISION (OPT = 0/4)
C*****************************
C
C*
C**  PERFORM UP TO FOUR (STANDARD) SUBDIVISIONS
C*
C*  FIND LOCATION OF CENTER OF FIRST AREA (SECTION)
C
      CTRLON=(LONNW+LONSW+LONSE+LONNE)/4.
      CTRLAT=(LATNW+LATSW+LATSE+LATNE)/4.
C
C*  DO NOTHING IF INDICATED, ELSE PERFORM NSUB SUBDIVISIONS, STARTING
C*  WITH SBD(1), EACH APPLIED TO THE AREA CREATED BY THE PREVIOUS ONES,
C*  AND THE TYPE OF EACH INDICATED BY THE CORRESPONDING CODE DIGIT IN
C*  ORDER.  SBD(1) WITH THE LEAST SIGNIFICANT DIGIT, AND SO ON
C
      IF (ORDER.EQ.0) GOTO 510
      DO 500 I=1,NSUB
        NUM = MOD ( INT (ORDER), 10)
C
C     **  REMOVE THE DIGIT FOR THIS SUBDIVISION
C
        ORDER = ORDER / 10
        IF (NUM.EQ.0) GOTO 510
C
C*********************************************
C * * * * * * * * * * * * * * * * * * * * * * QUARTER-AREA SUBDIVISION
C*********************************************
C
C     **  FIRST THE 1/4 AREAS
C
        IF (NUM.EQ.1) THEN
C
C     **  SOUTH EAST
C
        LONSW=(LONSW+LONSE)/2.
        LATSW=(LATSW+LATSE)/2.
        LONNE=(LONNE+LONSE)/2.
```

```
      LATNE=(LATNE+LATSE)/2.
      LONNW=CTRLON
      LATNW=CTRLAT
      ELSEIF (NUM.EQ.2) THEN
C
C     ** NORTH EAST
C
      LONSE=(LONNE+LONSE)/2.
      LATSE=(LATNE+LATSE)/2.
      LONNW=(LONNW+LONNE)/2.
      LATNW=(LATNW+LATNE)/2.
      LONSW=CTRLON
      LATSW=CTRLAT
      ELSEIF (NUM.EQ.3) THEN
C
C     ** NORTH WEST
C
      LONSW=(LONSW+LONNW)/2.
      LATSW=(LATSW+LATNW)/2.
      LONNE=(LONNW+LONNE)/2.
      LATNE=(LATNW+LATNE)/2.
      LONSE=CTRLON
      LATSE=CTRLAT
      ELSEIF (NUM.EQ.4) THEN
C
C     ** SOUTH WEST
C
      LONNW=(LONNW+LONSW)/2.
      LATNW=(LATNW+LATSW)/2.
      LONSE=(LONSW+LONSE)/2.
      LATSE=(LATSW+LATSE)/2.
      LONNE=CTRLON
      LATNE=CTRLAT
C
C*************************************************
C ************************ HALF-AREA SUBDIVISION
C*************************************************
C
C     ** NOW FOR THE 1/2 AREA SUBDIVISIONS
C
      ELSEIF (NUM.EQ.5) THEN
C
C     ** NORTH
C
      LONSW=(LONNW+LONSW)/2.
      LATSW=(LATNW+LATSW)/2.
      LONSE=(LONSE+LONNE)/2.
      LATSE=(LATSE+LATNE)/2.
      ELSEIF (NUM.EQ.6) THEN
C
C     ** WEST
C
      LONNE=(LONNE+LONNW)/2.
      LATNE=(LATNE+LATNW)/2.
      LONSE=(LONSE+LONSW)/2.
      LATSE=(LATSE+LATSW)/2.
      ELSEIF (NUM.EQ.7) THEN
C
C     ** SOUTH
```

```
C
      LONNW=(LONNW+LONSW)/2.
      LATNW=(LATNW+LATSW)/2.
      LONNE=(LONNE+LONSE)/2.
      LATNE=(LATNE+LATSE)/2.
      ELSEIF (NUM.EQ.8) THEN
C
C     ** EAST
C
      LONSW=(LONSW+LONSE)/2.
      LATSW=(LATSW+LATSE)/2.
      LONNW=(LONNW+LONNE)/2.
      LATNW=(LATNW+LATNE)/2.
      ELSE
      STAT=-4-I
      RETURN
      ENDIF
C
C     ** CALCULATE THE NEW CENTER
C
      CTRLON=(LONNW+LONNE+LONSE+LONSW)/4.
      CTRLAT=(LATNW+LATNE+LATSE+LATSW)/4.
500     CONTINUE
C
C
C*********************************************
C* * * * * * * * * * * * * * * * * * * * * * APPLICATION OF CORNR SPEC
C* * * * * * * * * * * * * * * * * * * * * * * * * (OPT = 0/4, 1/5)
C*********************************************
C
C
C     * IF CORNR OPTION IS USED,SELECT THE INDICATED POINT OF THE
C     * NINE SIGNIFICANT POINTS IN THE TARGET AREA, OR IF BLANK
C     * USE THE CENTER OF THE AREA AS THE SELECTED POINT
C
510     IF (CORNR.EQ.' ') THEN
      PLON=CTRLON
      PLAT=CTRLAT
C
C
C* * * * * * * * * * * * * * * * * * * * * * * * FOUR CORNERS OF A SUBAREA
C
C
C     ** IF CORNR SPEC IS BLANK AND OUTPUT IS NOT IN DMS, ALSO
C     ** PUT THE CORNER LOCATIONS OF THE TARGET AREA INTO THE
C     ** LEOXTR VARIABLES (LONX, LATX) IN DECIMAL DEGREES
C
      IF (OPT.LT.2) THEN
      LONX(1,1) = ABS(LONNW)
      LONX(2,1) = ABS(LONNE)
      LONX(3,1) = ABS(LONSW)
      LONX(4,1) = ABS(LONSE)
      LATX(1,1) = LATNW
      LATX(2,1) = LATNE
      LATX(3,1) = LATSW
      LATX(4,1) = LATSE
      ENDIF
C
C* * * * * * * * * * * * * * * * * * * * * * * APPLYING THE CORNR SPEC
```

```
C
C
      ELSEIF (CORNR.EQ.'NE' .OR. CORNR.EQ.'A ') THEN
        PLON=LONNE
        PLAT=LATNE
      ELSEIF (CORNR.EQ.'NE' .OR. CORNR.EQ.'A ') THEN
        PLON=LONNE
        PLAT=LATNE
      ELSEIF (CORNR.EQ.'NW' .OR. CORNR.EQ.'B ') THEN
        PLON=LONNW
        PLAT=LATNW
      ELSEIF (CORNR.EQ.'NW' .OR. CORNR.EQ.'B ') THEN
        PLON=LONNW
        PLAT=LATNW
      ELSEIF (CORNR.EQ.'SW' .OR. CORNR.EQ.'C ') THEN
        PLON=LONSW
        PLAT=LATSW
      ELSEIF (CORNR.EQ.'SW' .OR. CORNR.EQ.'C ') THEN
        PLON=LONSW
        PLAT=LATSW
      ELSEIF (CORNR.EQ.'SE' .OR. CORNR.EQ.'D ') THEN
        PLON=LONSE
        PLAT=LATSE
      ELSEIF (CORNR.EQ.'SE' .OR. CORNR.EQ.'D ') THEN
        PLON=LONSE
        PLAT=LATSE
      ELSEIF (CORNR.EQ.'S ' .OR. CORNR.EQ.'S ') THEN
        PLON=(LONSW+LONSE)/2.
        PLAT=(LATSW+LATSE)/2.
      ELSEIF (CORNR.EQ.'E ' .OR. CORNR.EQ.'E ') THEN
        PLON=(LONSE+LONNE)/2.
        PLAT=(LATSE+LATNE)/2.
      ELSEIF (CORNR.EQ.'N ' .OR. CORNR.EQ.'N ') THEN
        PLON=(LONNW+LONNE)/2.
        PLAT=(LATNW+LATNE)/2.
      ELSEIF (CORNR.EQ.'W ' .OR. CORNR.EQ.'W ') THEN
        PLON=(LONNW+LONSW)/2.
        PLAT=(LATNW+LATSW)/2.
      ELSE
        STAT=-25
        RETURN
      ENDIF
C
C     * POINT LONGITUDE AND LATITUDE HAVE BEEN IDENTIFIED,NOW
C     * TRANSLATE TO DMS IF REQUIRED,SET STAT VALUE AND EXIT
C
C
C*****************************************
C * * * * * * * * * * * * * * * * * * * * DMS CONVERSION (OPT = 4/5/6)
C*****************************************
C
C
C     * FOR THE DMS OPTION CONVERT TO DEGREES-MINUTES-SECONDS
C
550       IF (OPT.GT.3) THEN
        PLON=ABS(PLON)
        LONX(1,1)=INT (PLON)
        LATX(1,1)=INT (PLAT)
        LONX(1,2)=60.*(PLON-LONX(1,1))
```

```
       LATX(1,2)=60.*(PLAT-LATX(1,1))
       LONX(1,3)=60.*(LONX(1,2)-INT (LONX(1,2)))
       LATX(1,3)=60.*(LATX(1,2)-INT (LATX(1,2)))
       LONX(1,2)=INT (LONX(1,2))
       LATX(1,2)=INT (LATX(1,2))
C
C      * CHECK FOR AND CORRECT OVERFLOW ERRORS
C
       IF (LONX(1,3).GE.60.) THEN
      LONX(1,3)=LONX(1,3)-60.
      LONX(1,2)=LONX(1,2)+1.
       ENDIF
       IF (LONX(1,2).GE.60.) THEN
      LONX(1,2)=LONX(1,2)-60.
      LONX(1,1)=LONX(1,1)+1.
       ENDIF
       IF (LATX(1,3).GE.60.) THEN
       LATX(1,3)=LATX(1,3)-60.
       LATX(1,2)=LATX(1,2)+1.
       ENDIF
       IF (LATX(1,2).GE.60.) THEN
       LATX(1,2)=LATX(1,2)-60.
       LATX(1,1)=LATX(1,1)+1.
       ENDIF
     ENDIF
C
C*********************************************
C* * * * * * * * * * * * * * * * * * * * * * SUCCESS, SET STAT = 1
C*********************************************
C
C      * SUCCESS,SET STAT=1.  IF THIS IS IN A SECTION THAT IS
C      * MARKED AS NON-STANDARD (SFLAG > 0),NONSTD=1 ELSE 0
C
575       STAT=1
     IF (SFLAG(SECT).EQ.0) THEN
       NONSTD=0
     ELSE
       NONSTD=1
     ENDIF
C
C      * MAKE CERTAIN THE LONGITUDE IS POSITIVE AND RETURN
C
     IF (OPT.LT.3) PLON=ABS(PLON)
     RETURN
   ENDIF
C
C*
C    ** NOW FOR THE SECOND OPTION SORCE=2,IN WHICH THE SORCE
C    ** INFORMATION IS GEOGRAPHIC (LONGITUDE,LATITUDE) AND THE
C    ** CALLING ROUTINE WANTS THE LEGAL (TOWNSHIP/RANG/SECTION)
C    ** DESCRIPTION TO BE RETURNED
C
C**
C*************************************************************************
C* * * * * * * * * * * * * * * * * * * * GEOGRAPHIC TO LEGAL CONVERSION
C*************************************************************************
C**
C
C      * FIRST MAKE AN EDUCATED GUESS ON THE TOWNSHIP.  THE RANG
```

```
C     * NUMBER HERE IS ABSOLUTE WITH R43W BEING 1 AND R25E IS 68
C
      TROW=(BASLAT-PLAT)/DELAT
      TROW=TROW+1
      I=(TROW-1)/30+1
      TCOL=(PLON-WSTLON(I))/DELON(I)
      TCOL=TCOL+1
      TSHP=(TROW-1)/6+1
      RANG=(TCOL-1)/6+1
C
C     * CALCULATE THE RECORD NUMBER FOR THE TOWNSHIP, CHECK IF IT HAS
C     * ALREADY BEEN ACCESSED ("THRASHING") AND THEN READ THE RECORD
C
      RTOP=1
600   TREC=(TSHP-1)*68+RANG
      IF (TREC.LT.0 .OR. TREC.GT.2380) THEN
        STAT = -27
        GOTO 1500
      ENDIF
      IF (VZT(TREC).LT.2) THEN
        READ(LUN,REC=TREC,ERR=1999)
     &    ((LON(I,J),LAT(I,J),J=1,7),I=1,7), TFLAG, SFLAG
        IF (TRCE) WRITE(*,*)
     &      '** VISIT TSHP,RANG= ',TSHP,RANG,' FLAG= ',TFLAG,' **'
        VZT(TREC)=VZT(TREC)+1
        VZTD(RTOP)=TREC
        RTOP=RTOP+1
C
C* CHECK TOWNSHIP FOR LACK OF DATA
C
        IF (TFLAG.EQ.-1) THEN
          STAT=-19
          GOTO 1500
        ENDIF
C
C     * IF THIS IS 2ND VISIT, SKIP TSHP ANALYSIS - FIND SECTION
C

        IF (VZT(TREC).GT.1) THEN
          IF (TRCE) WRITE(*,*)'** PREVIOUSLY VISITED TSHP **'
          GOTO 700
        ENDIF
C
C     * IF SOME CORNERS ARE MISSING, GO DO SECTION SEARCH
C
        IF (TFLAG.GT.0) THEN
          IF (TRCE) WRITE(*,*)'**  PARTIALLY FULL TOWNSHIP  **'
          GOTO 700
        ENDIF
      ELSE
C
C     * THRASHING HAS OCCURRED IN THE TOWNSHIP SEARCH
C
        STAT=-14
        GOTO 1500
      ENDIF
C
C
C*********************************************************
```

```
C**************************** TOWNSHIP SEARCH
C*****************************************************
C
C
C     * THE FIRST SEARCHES WILL BE TO LOCATE THE POINT AS BEING
C     * EASILY OUTSIDE OR INSIDE THE TOWNSHIP.  FOR THIS,TWO
C     * BOXES ARE USED. THE FIRST IS THE OUTER BOX,DEFINED AS
C     * THE MINIMUM BOUNDING RECTANGLE DEFINED BY THE MINIMUM
C     * AND MAXIMUM COORDINATES OF THE 4 CORNERS.  THE OTHER BOX
C     * IS THE INNER BOX,DEFINED BY THE COORDINATES NOT USED
C     * FOR THE OUTER BOX.  IF THE POINT IS OUTSIDE THE OUTER
C     * BOX,THEN IT IS NOT IN THE TOWNSHIP.  LIKEWISE,IF IT IS
C     * INSIDE THE INNER BOX,IT IS INSIDE THE TOWNSHIP.
C***
C*****************************************
C********************** SEARCH INTERIOR AND EXTERIOR
C************************ RECTANGLES (EASY SEARCH)
C*****************************************
C
C     * SO NOW FIND THE TWO BOXES
C
   IF (LON(1,1).LT.LON(7,1)) THEN
     LONOMN=LON(1,1)
     LONIMN=LON(7,1)
   ELSE
     LONOMN=LON(7,1)
     LONIMN=LON(1,1)
   ENDIF
   IF (LAT(1,1).GT.LAT(1,7)) THEN
     LATOMX=LAT(1,1)
     LATIMX=LAT(1,7)
   ELSE
     LATOMX=LAT(1,7)
     LATIMX=LAT(1,1)
   ENDIF
   IF (LON(1,7).GT.LON(7,7)) THEN
     LONOMX=LON(1,7)
     LONIMX=LON(7,7)
   ELSE
     LONOMX=LON(7,7)
     LONIMX=LON(1,7)
   ENDIF
   IF (LAT(7,7).LT.LAT(7,1)) THEN
     LATOMN=LAT(7,7)
     LATIMN=LAT(7,1)
   ELSE
     LATOMN=LAT(7,1)
     LATIMN=LAT(7,7)
   ENDIF
C
C     ** LOOK FOR POINT OUTSIDE THE OUTER BOX.  IF FOUND,RESET
C     ** TSHP AND/OR RANG TO SEARCH THE NEXT TOWNSHIP
C
   OUTFLG=0
   IF (PLON.GT.LONOMX) THEN
     OUTFLG=1
     RANG=RANG+1
   ELSEIF (PLON.LT.LONOMN) THEN
     OUTFLG=1
```

```
      RANG=RANG-1
    ENDIF
    IF (PLAT.GT.LATOMX) THEN
      OUTFLG=1
      TSHP=TSHP-1
    ELSEIF (PLAT.LT.LATOMN) THEN
      OUTFLG=1
      TSHP=TSHP+1
    ENDIF
    IF(TRCE .AND. OUTFLG.NE.0)THEN
      WRITE(*,*)'** OUTSIDE T-BOX, NEW TSHP,RANG= ',
    &         TSHP,RANG,' **'
      WRITE(*,*)'   PLON, PLAT = ',PLON,PLAT
      WRITE(*,*)'++ OUTERBOX AT: ',LONOMN,LATOMN,LONOMX,LATOMX
    ENDIF
    IF (OUTFLG.NE.0) GOTO 600
C
C     * CHECK THE INNER BOX FOR POINTS EASILY INSIDE.  WHEN FOUND,
C     * GO TO SEARCH FOR THE SECTION WITHIN THE TOWNSHIP
C
    OUTFLG=0
    IF (PLON.LT.LONIMN) THEN
      OUTFLG=1
    ELSEIF (PLON.GT.LONIMX) THEN
      OUTFLG=4
    ENDIF
    IF (PLAT.LT.LATIMN) THEN
      OUTFLG=OUTFLG+2
    ELSEIF (PLAT.GT.LATIMX) THEN
      OUTFLG=OUTFLG+8
    ENDIF
    IF (OUTFLG.LT.1) THEN
      IF (TRCE) WRITE (*,*) '**  INSIDE TSHP BOX  **'
      GOTO 700
    ENDIF
C
C*
C     ** THESE POINTS WERE NOT ELIMINATED BY THE EASY METHOD,SO
C     ** CHECK THE POINT AGAINST THE FOUR SIDES AS FOLLOWS.  IF THE
C     ** POINT IS OUTSIDE THE ENDPOINTS OF THE SIDE INDICATED BY
C     ** THE ABOVE EVALUATION,THEN SEND IT OFF TO ANOTHER TOWNSHIP.
C     ** OTHERWISE,FIND THE INTERSECTION POINT OF A VERTICAL OR
C     ** HORIZONTAL LINE THROUGH THE POINT WITH THE SIDE.  IF THIS
C     ** INTERSECTION POINT IS INSIDE THE POINT,THEN THE POINT IS
C     ** OUTSIDE THE TOWNSHIP.
C
C*****************************************
C * * * * * * * * * * * * * * * * * * * * *SEARCH BETWEEN INTERIOR AND
C* * * * * * * * * * * * * * * * * * * * EXTERIOR RECTANGLES (COMPLEX)
C*****************************************
C
    IF (TRCE) WRITE (*,*) '**  FANCY TSHP TESTS  **'
    IF (OUTFLG.GT.7) THEN
C
C     ** HERE TO TEST THOSE ABOVE THE NORTH SIDE
C
      OUTFLG=OUTFLG-8
      IF (PLON.LT.LON(1,1)) THEN
        TSHP=TSHP-1
```

```
        RANG=RANG-1
        GOTO 600
      ELSEIF (PLON.GT.LON(1,7)) THEN
        TSHP=TSHP-1
        RANG=RANG+1
        GOTO 600
      ENDIF
C
C     ** FIND INTERSECTION OF VERTICAL THROUGH POINT WITH NORTH SIDE
C
      DLAT=(LAT(1,7)-LAT(1,1))*(PLON-LON(1,1))
      DLAT=DLAT/(LON(1,7)-LON(1,1))
      XLAT=LAT(1,1)+DLAT
      IF (PLAT.GT.XLAT) THEN
        TSHP=TSHP-1
        GOTO 600
      ELSE
        GOTO 700
      ENDIF
    ENDIF
    IF (OUTFLG.GT.3) THEN
C
C     ** NOW FOR THE EAST SIDE TESTS
C
      OUTFLG=OUTFLG-4
C
C     ** IS THE POINT ABOVE OR BELOW THE SIDE?
C
      IF (PLAT.GT.LAT(1,7)) THEN
        TSHP=TSHP-1
        RANG=RANG+1
        GOTO 600
      ELSEIF (PLAT.LT.LAT(7,7)) THEN
        TSHP=TSHP+1
        RANG=RANG+1
        GOTO 600
      ENDIF
C
C     ** NO, FIND INTERSECTION WITH HORIZONTAL
C
      DLON=(LON(7,7)-LON(1,7))*(PLAT-LAT(1,7))
      DLON=DLON/(LAT(7,7)-LAT(1,7))
      XLON=LON(1,7)+DLON
      IF (PLON.GT.XLON) THEN
        RANG=RANG+1
        GOTO 600
      ELSE
        GOTO 700
      ENDIF
    ENDIF
    IF (OUTFLG.GT.1) THEN
C
C     ** NOW TESTING FOR THE SOUTH SIDE
C
      OUTFLG=OUTFLG-2
C
C     ** IS POINT BELOW THE BOTTOM SIDE?
C
      IF (PLON.LT.LON(7,1)) THEN
```

```
        TSHP=TSHP+1
        RANG=RANG-1
        GOTO 600
      ELSEIF (PLON.GT.LON(7,7)) THEN
        TSHP=TSHP+1
        RANG=RANG+1
        GOTO 600
      ENDIF
C
C     ** FIND INTERSECTION POINT WITH VERTICAL AND TEST
C
      DLAT=(LAT(7,1)-LAT(7,7))*(PLON-LON(7,1))
      DLAT=DLAT/(LON(7,7)-LON(7,1))
      XLAT=LAT(7,1)+DLAT
      IF (PLAT.LT.XLAT) THEN
        TSHP=TSHP+1
        GOTO 600
      ELSE
        GOTO 700
      ENDIF
    ENDIF
    IF (OUTFLG.GT.0) THEN
C
C     ** TEST THE WEST SIDE
C
      OUTFLG=0
C
C     ** IS POINT OUTSIDE THE SEGMENT?
C
      IF (PLAT.GT.LAT(1,1)) THEN
        TSHP=TSHP-1
        RANG=RANG-1
        GOTO 600
      ELSEIF (PLAT.LT.LAT(7,1)) THEN
        TSHP=TSHP+1
        RANG=RANG-1
        GOTO 600
      ENDIF
C
C     ** FIND HORIZONTAL INTERSECTION AND CHECK
C
      DLON=(LON(1,1)-LON(7,1))*(PLAT-LAT(7,1))
      DLON=DLON/(LAT(1,1)-LAT(7,1))
      XLON=LON(7,1)+DLON
      IF (PLON.LT.XLON) THEN
        RANG=RANG-1
        GOTO 600
      ENDIF
    ENDIF
C
C
C************************************
C * * * * * * * * * * * * * * * * * TOWNSHIP FOUND, SEARCH FOR SECTION
C************************************
C
C***
C     THE POINT IS (PROBABLY) INSIDE THIS TOWNSHIP, AT LEAST
C     ACCORDING TO THE FOUR TOWNSHIP CORNERS.  THE ACTUAL
C     SECTION BOUNDARIES MAY NOT ALWAYS AGREE.  ANYWAY,NOW
```

```
C     THE MOST LIKELY SECTION IS SELECTED ACCORDING TO WORLD
C     KNOWLEDGE OF THE SIZE OF SECTIONS IN VARIOUS PARTS OF
C     THE STATE. THE SECTION IS SEARCHED IN THE THOROUGH
C     MANNER AS THE TOWNSHIPS WERE. IF THE POINT IS FOUND TO
C     LIE OUTSIDE THE SECTION,A NEW TARGET SECTION IS TRIED.
C     IN THIS PROCESS,IT IS POSSIBLE TO MOVE TO A DIFFERENT
C     TARGET TOWNSHIP. WHEN THE CORRECT SECTION IS FOUND,
C     THE SUBDIVISIONS ARE GENERATED FOR THE COMPLETE LEGAL
C     DESCRIPTION AND THE TASK IS COMPLETE.
C***
C
C     * ZERO THE SEARCH MASK (SEARCH) TO INDICATE THAT NO SECTIONS
C     * HAVE BEEN SEARCHED FOR THIS TOWNSHIP. AS THEY
C     * ARE SEARCHED, THE VALUE IS SET TO 1 TO PREVENT THRASHING.
C
700    DO 725 I=1,6
       DO 725 J=1,6
         SEARCH(I,J)=0
725    CONTINUE
   IF (TFLAG.EQ.0) THEN
C
C     ** NOW TO SELECT THE INITIAL SECTION FOR THE SEARCH
C
C     ** FOR FULL TOWNSHIPS CALCULATE THE ROW AND COLUMN
C     ** OF THE NW CORNER OF THE MOST LIKELY SECTION,
C     ** BASED ON THE SPATIAL RELATIONSHIPS OF THE CORNERS
C
     SROW=6*(LAT(1,1)-PLAT)/(LAT(1,1)-LAT(7,1))
     SCOL=6*(PLON-LON(1,1))/(LON(1,7)-LON(1,1))
     SROW=SROW+1
     SCOL=SCOL+1
     IF (TRCE) WRITE(*,*)'** INITIAL SROW, SCOL = ',SROW,SCOL
   ELSE
C
C     ** FOR TOWNSHIPS WITH FEWER THAN 36 COMPLETE SECTIONS
C     ** ALL SECTIONS WILL BE CHECKED IF NECESSARY. START
C     ** IN THE NW CORNER (SECTION 6) AND SEARCH SECTIONS
C     ** THAT HAVE 4 CORNERS UNTIL A SECTION CONTAINS THE
C     ** POINT. IF NONE DOES, END WITH AN ERROR.
C
     SROW = 1
     SCOL = 1
     GOTO 775
   ENDIF
C
C     * MAKE SURE THE SECTION IS IN THIS TOWNSHIP. THIS IS MORE
C     * IMPORTANT WHEN RECYCLING FOR A NEIGHBORING SECTION. (IF
C     * NECESSARY,GO SEARCH THE NEIGHBORING TOWNSHIP.)
C
750    OUTFLG=0
   IF (SROW.LT.1) THEN
     OUTFLG=1
     TSHP=TSHP-1
   ELSEIF (SROW.GT.6) THEN
     OUTFLG=1
     TSHP=TSHP+1
   ENDIF
   IF (SCOL.LT.1) THEN
     OUTFLG=1
```

```
      RANG=RANG-1
    ELSEIF (SCOL.GT.6) THEN
      OUTFLG=1
      RANG=RANG+1
    ENDIF
    IF (OUTFLG.GT.0) THEN
      OUTFLG=0
      GOTO 600
    ENDIF
C
C     * SET POINTERS TO CORNERS OF SECTION. IF SECTION HAS NOT
C     * BEEN SEARCHED,CHECK FOR UNKNOWN VALUES IN LEO2BASE.
C
775   N=SROW
    S=SROW+1
    W=SCOL
    E=SCOL+1
C
C     ** IF SECTION HAS ALREADY BEEN SEARCHED-A THRASHING ERROR.
C     ** OTHERWISE,MARK THE SECTION AS HAVING BEEN SEARCHED.
C
    IF (TFLAG.EQ.0 .AND. SEARCH(N,W).NE.0) THEN
      STAT=-15
      GOTO 1500
    ELSE
      SEARCH(N,W)=SEARCH(N,W)+1
    ENDIF
C
C     * CALCULATE THE SECTION NUMBER FROM ROW AND COLUMN
C     * CHECK FOR MISSING CORNER VALUES VIA THE SECTION FLAG
C
    IF (MOD (INT (SROW),2).EQ.0) THEN
      SECT=6*(SROW-1)+SCOL
    ELSE
      SECT=6*SROW+1-SCOL
    ENDIF
    IF (TRCE) WRITE (*,*) '* * *  SECTION = ', SECT
C
C     * FOR PARTIAL TOWNSHIPS, JUST PROCEED THROUGH THE SECTIONS
C
    IF (TFLAG.GT.0) THEN
      IF (SECT.GT.35) THEN
        STAT = -28
        GOTO 1500
      ENDIF
C
C     * SECTIONS WITH 4 CORNERS MAY BE SEARCHED
C
      IF (SFLAG(SECT).GT.-1) GOTO 780
C
C     * TRY THE NEXT SECTION
C
      SCOL = SCOL + 1
      IF (SCOL.GT.6) THEN
        SCOL = 1
        SROW = SROW + 1
        IF (SROW.GT.6) THEN
        STAT = -28
        GOTO 1500
```

```
        ENDIF
      ENDIF
      GOTO 775
    ENDIF
    IF (SFLAG(SECT).LT.0) THEN
      STAT=SFLAG(SECT)-20
      GOTO 1500
    ENDIF
C
C*************************** SIMPLE SECTION SEARCH
C
C********* CREATE INNER AND OUTER BOXES FOR SIMPLE CASES
C
780   IF (LON(N,W).LT.LON(S,W)) THEN
      LONOMN=LON(N,W)
      LONIMN=LON(S,W)
    ELSE
      LONOMN=LON(S,W)
      LONIMN=LON(N,W)
    ENDIF
    IF (LAT(N,W).GT.LAT(N,E)) THEN
      LATOMX=LAT(N,W)
      LATIMX=LAT(N,E)
    ELSE
      LATOMX=LAT(N,E)
      LATIMX=LAT(N,W)
    ENDIF
    IF (LON(N,E).GT.LON(S,E)) THEN
      LONOMX=LON(N,E)
      LONIMX=LON(S,E)
    ELSE
      LONOMX=LON(S,E)
      LONIMX=LON(N,E)
    ENDIF
    IF (LAT(S,E).LT.LAT(S,W)) THEN
      LATOMN=LAT(S,E)
      LATIMN=LAT(S,W)
    ELSE
      LATOMN=LAT(S,W)
      LATIMN=LAT(S,E)
    ENDIF
C
C     * LOOK FOR POINT OUTSIDE THE OUTER BOX.  IF FOUND, RESET
C     * THE SROW AND/OR SCOL POINTERS FOR THE NEXT SEARCH.
C
    OUTFLG=0
    IF (PLON.GT.LONOMX) THEN
      OUTFLG=1
      SCOL=SCOL+1
    ELSEIF (PLON.LT.LONOMN) THEN
      OUTFLG=1
      SCOL=SCOL-1
    ENDIF
    IF (PLAT.GT.LATOMX) THEN
      OUTFLG=1
      SROW=SROW-1
    ELSEIF (PLAT.LT.LATOMN) THEN
      OUTFLG=1
      SROW=SROW+1
```

```
      ENDIF
C
C     * IF POINT IS OUT (OUTFLG=1),GO FOR NEXT SECTION
C
      IF (OUTFLG.NE.0) THEN
        IF(TRCE) WRITE(*,*)
     &        '* OUTSIDE SECTION BOX, SROW, SCOL= ',SROW,SCOL,' *'
        GOTO 750
      ENDIF
C
C     ** CHECK THE INNER BOX FOR POINTS EASILY INSIDE
C
      OUTFLG=0
      IF (PLON.LT.LONIMN) THEN
        OUTFLG=1
      ELSEIF (PLON.GT.LONIMX) THEN
        OUTFLG=4
      ENDIF
      IF (PLAT.LT.LATIMN) THEN
        OUTFLG=OUTFLG+2
      ELSEIF (PLAT.GT.LATIMX) THEN
        OUTFLG=OUTFLG+8
      ENDIF

      IF (OUTFLG.LT.1) THEN
        IF (TRCE) WRITE (*,*) '* * *  INSIDE SECTION BOX  * * *'
        GOTO 800
      ENDIF
C
C*********************************************
C * * * * * * * * * * * * * * * * * * * * * * (COMPLEX) SEARCH BETWEEN
C* * * * * * * * * * * * * * * * * * * * * * * INNER AND OUTER BOXES
C*********************************************
C*
C     * THESE POINTS WERE NOT ELIMINATED BY THE EASY METHOD,SO
C     * CHECK THE POINT AGAINST THE FOUR SIDES AS FOLLOWS.  IF THE
C     * POINT IS OUTSIDE THE ENDPOINTS OF THE SIDE INDICATED BY
C     * THE ABOVE EVALUATION,THEN SEND IT OFF TO ANOTHER TOWNSHIP.
C     * OTHERWISE,FIND THE INTERSECTION POINT OF A VERTICAL OR
C     * HORIZONTAL LINE THROUGH THE POINT WITH THE SIDE.  IF THIS
C     * INTERSECTION POINT IS INSIDE THE POINT,THEN THE POINT IS
C     * OUTSIDE THE TOWNSHIP.
C**************
C
      IF (TRCE) WRITE (*,*)
     &        '* * FANCY SECTION TESTS, OUTFLG= ',OUTFLG,' * *'
      IF (OUTFLG.GT.7) THEN
C
C     ** HERE TO TEST THOSE ABOVE THE NORTH SIDE
C
        OUTFLG=OUTFLG-8
        IF (PLON.LT.LON(N,W)) THEN
          SROW=SROW-1
          SCOL=SCOL-1
          GOTO 750
        ELSEIF (PLON.GT.LON(N,E)) THEN
          SROW=SROW-1
          SCOL=SCOL+1
          GOTO 750
```

```
      ENDIF
C
C     * FIND INTERSECTION OF VERTICAL THROUGH POINT WITH TOP
C
      DLAT=(LAT(N,E)-LAT(N,W))*(PLON-LON(N,W))
      DLAT=DLAT/(LON(N,E)-LON(N,W))
      XLAT=LAT(N,W)+DLAT
      IF (PLAT.GT.XLAT) THEN
        SROW=SROW-1
        GOTO 750
      ELSE
        GOTO 800
      ENDIF
    ENDIF
    IF (OUTFLG.GT.3) THEN
C
C     ** NOW FOR THE EAST SIDE TESTS
C
      OUTFLG=OUTFLG-4
C
C     * IS THE POINT ABOVE OR BELOW THE SIDE?
C
      IF (PLAT.GT.LAT(N,E)) THEN
        SROW=SROW-1
        SCOL=SCOL+1
        GOTO 750
      ELSEIF (PLAT.LT.LAT(S,E)) THEN
        SROW=SROW+1
        SCOL=SCOL+1
        GOTO 750
      ENDIF
C
C     * NO, FIND THE INTERSECTION WITH HORIZONTAL
C
      DLON=(LON(S,E)-LON(N,E))*(PLAT-LAT(N,E))
      DLON=DLON/(LAT(S,E)-LAT(N,E))
      XLON=LON(N,E)+DLON
      IF (PLON.GT.XLON) THEN
        SCOL=SCOL+1
        GOTO 750
      ELSE
        GOTO 800
      ENDIF
    ENDIF
    IF (OUTFLG.GT.1) THEN
C
C     ** NOW TESTING FOR THE SOUTH SIDE
C
      OUTFLG=OUTFLG-2
C
C     * IS THE POINT OUTSIDE THE SOUTH SIDE?
C
      IF (PLON.LT.LON(S,W)) THEN
        SROW=SROW+1
        SCOL=SCOL-1
        GOTO 750
      ELSEIF (PLON.GT.LON(S,E)) THEN
        SROW=SROW+1
        SCOL=SCOL+1
```

```
        GOTO 750
      ENDIF
C
C     * CALCULATE INTERSECTION POINT WITH VERTICAL AND TEST
C
      DLAT=(LAT(S,W)-LAT(S,E))*(PLON-LON(S,W))
      DLAT=DLAT/(LON(S,E)-LON(S,W))
      XLAT=LAT(S,W)+DLAT
      IF (PLAT.LT.XLAT) THEN
        SROW=SROW+1
        GOTO 750
      ELSE
        GOTO 800
      ENDIF
    ENDIF
    IF (OUTFLG.GT.0) THEN
C
C     ** TEST THE WEST SIDE
C
      OUTFLG=0
C
C     * IS POINT OUTSIDE THE SEGMENT?
C
      IF (PLAT.GT.LAT(N,W)) THEN
        SROW=SROW-1
        SCOL=SCOL-1
        GOTO 750
      ELSEIF (PLAT.LT.LAT(S,W)) THEN
        SROW=SROW+1
        SCOL=SCOL-1
        GOTO 750
      ENDIF
C
C     * FIND HORIZONTAL INTERSECTION AND CHECK
C
      DLON=(LON(N,W)-LON(S,W))*(PLAT-LAT(S,W))
      DLON=DLON/(LAT(N,W)-LAT(S,W))
      XLON=LON(S,W)+DLON
      IF (PLON.LT.XLON) THEN
        SCOL=SCOL-1
        GOTO 750
      ENDIF
    ENDIF
C
C**************************************
C* * * * * * * * * * * * * * * * * * * POINT FOUND - SET SUBDIVISIONS
C* * * * * * * * * * * * * * * * * * * AND CLOSEST OF 9 POINTS (CORNR)
C**************************************
C
C     THE POINT IS INSIDE THIS SECTION!!!
C
C     APPLY UP TO FOUR SUBDIVISIONS AND EXIT.
C
C     ** GET COORDINATES OF CORNERS FOR SUBDIVISION
C
800   LONSW=LON(S,W)
    LATSW=LAT(S,W)
    LONSE=LON(S,E)
    LATSE=LAT(S,E)
```

```
      LONNE=LON(N,E)
      LATNE=LAT(N,E)
      LONNW=LON(N,W)
      LATNW=LAT(N,W)
      IF (TRCE) WRITE (*,*) '* * *  POINT INSIDE SECTION !!  * * *'
C
C*************
C     * SUBDIVIDE EACH RECTANGLE ACCORDING TO THE QUARTER THAT
C     * THE POINT LIES IN (NW,SW,SE,NE) ONCE FOR EACH OF THE
C     * FOUR SUBDIVISIONS THAT ARE SET ALONG THE WAY (SBD).
C*************
C
      LEVEL=1
C
C* FIND SIDE MIDPOINT AND CENTER LOCATIONS
C
1000  LONN=(LONNW+LONNE)/2.
      LATN=(LATNW+LATNE)/2.
      LONE=(LONNE+LONSE)/2.
      LATE=(LATNE+LATSE)/2.
      LONS=(LONSE+LONSW)/2.
      LATS=(LATSE+LATSW)/2.
      LONW=(LONSW+LONNW)/2.
      LATW=(LATSW+LATNW)/2.
      CTRLON=(LONSW+LONSE+LONNE+LONNW)/4.
      CTRLAT=(LATSW+LATSE+LATNE+LATNW)/4.
      IF (LEVEL.GT.4) GOTO 1100
C
C     * TO SEE WHERE POINT IS,FIND LONGITUDE OF INTERSECTION OF
C     * HORIZONTAL LINE THROUGH POINT WITH LINE THROUGH NORTH AND
C     * SOUTH MIDPOINTS AND LATITUDE OF INTERSECTION OF VERTICAL
C     * LINE THROUGH POINT WITH LINE THROUGH EAST & WEST MIDPOINTS
C
      LONINT=LONN-(LONN-LONS)*(LATN-PLAT)/(LATN-LATS)
      LATINT=LATW+(LATE-LATW)*(PLON-LONW)/(LONE-LONW)
C
C     * SET THE 1/4-AREA SUBDIVISION FOR THE CURRENT LEVEL (LEVEL)
C
      IF (PLON.GT.LONINT) THEN
        IF (PLAT.GT.LATINT) THEN
          SBD(LEVEL)='NE'
          LONSW=CTRLON
          LATSW=CTRLAT
          LONNW=LONN
          LATNW=LATN
          LONSE=LONE
          LATSE=LATE
        ELSE
          SBD(LEVEL)='SE'
          LONSW=LONS
          LATSW=LATS
          LONNW=CTRLON
          LATNW=CTRLAT
          LONNE=LONE
          LATNE=LATE
        ENDIF
      ELSEIF (PLAT.GT.LATINT) THEN
        SBD(LEVEL)='NW'
        LONNE=LONN
```

```
      LATNE=LATN
      LONSE=CTRLON
      LATSE=CTRLAT
      LONSW=LONW
      LATSW=LATW
    ELSE
      SBD(LEVEL)='SW'
      LONNE=CTRLON
      LATNE=CTRLAT
      LONSE=LONS
      LATSE=LATS
      LONNW=LONW
      LATNW=LATW
    ENDIF
C
C     * CYCLE UNTIL 4 SUBDIVISIONS ARE SET
C
    IF (LEVEL.LT.5) THEN
      LEVEL=LEVEL+1
      GOTO 1000
    ENDIF
C
C*************
C     ** SET CORNR VALUE TO INDICATE CLOSEST OF NINE POINTS
C*************
C
1100   IF (PLON.LT.CTRLON) THEN
       IF (PLAT.LT.CTRLAT) THEN
         IF (CTRLON-PLON.GT.PLON-LONW) THEN
         IF (LATW-PLAT.GT.PLAT-LATSW) THEN
           CORNR='SW'
         ELSE
           CORNR='W '
         ENDIF
         ELSEIF (CTRLAT-PLAT.GT.PLAT-LATS) THEN
         CORNR='S '
         ELSE
         CORNR=' '
         ENDIF
       ELSE
         IF (CTRLON-PLON.GT.PLON-LONW) THEN
         IF (LATNW-PLAT.GT.PLAT-LATW) THEN
           CORNR='W '
         ELSE
           CORNR='NW'
         ENDIF
         ELSE
         IF (LATN-PLAT.GT.PLAT-CTRLAT) THEN
           CORNR=' '
         ELSE
           CORNR='N '
         ENDIF
         ENDIF
       ENDIF
     ELSEIF (PLAT.LT.CTRLAT) THEN
       IF (LONE-PLON.GT.PLON-CTRLON) THEN
         IF (CTRLAT-PLAT.GT.PLAT-LATS) THEN
         CORNR='S '
         ELSE
```

```
        CORNR=' '
        ENDIF
      ELSEIF (LATE-PLAT.GT.PLAT-LATSE) THEN
        CORNR='SE'
      ELSE
        CORNR='E '
      ENDIF
    ELSEIF (LONE-PLON.GT.PLON-CTRLON) THEN
      IF (LATN-PLAT.GT.PLAT-CTRLAT) THEN
        CORNR=' '
      ELSE
        CORNR='N '
      ENDIF
    ELSE
      IF (LATNE-PLAT.GT.PLAT-LATE) THEN
        CORNR='E '
      ELSE
        CORNR='NE'
      ENDIF
    ENDIF
C            ********************       ***
C**************** S U C C E S S ! ! !  **********
C            ********************       ***
    STAT=1
    IF (SFLAG(SECT).EQ.0) THEN
      NONSTD=0
    ELSE
      NONSTD=1
    ENDIF
C
C     * NOTHING LEFT TO DO BUT SET THE RANG INDICATORS TO THE
C     * EXPECTED E AND W VALUES,AND RESET THRASHING MONITOR
C     * ARRAYS TO ZERO FOR ANY SUBSEQUENT CALLS
C
    IF (RANG.LT.44) THEN
      RANG=44-RANG
      EW='W'
    ELSE
      RANG=RANG-43
      EW='E'
    ENDIF
C
C     * RESET THE THRASHING MONITORS IF NECESSARY
C
1500   IF (RTOP.GT.1) THEN
       DO 1600 I=1,RTOP-1
         VZT(VZTD(I))=0
         VZTD(I)=0
1600     CONTINUE
       RTOP=1
    ENDIF
    RETURN
C
C     * FOR ERR= ERRORS, SET ERROR FLAG AND RETURN
C
1999   STAT=-13
    IF (SORCE.EQ.2) GOTO 1500
    RETURN
    END
```