# NCGIA/U.S. Census Multiple Representations Data Set Project

# Technical Report on Pilot Project: Lee County, Florida

May, 1990

Joseph S. DeLotto - NCGIA Buffalo
Barbara P. Buttenfield - NCGIA Buffalo
Frederick Broome - U.S. Bureau of the Census

National Center for Geographic Information and Analysis

Report 90-4

## 1. Introduction

This report documents the joint effort of the NCGIA-Buffalo and the US Bureau of the Census to produce a multi-scale, multi-agency database intended for use in teaching and research. Additional commitments for data have come from the National Ocean Service, of NOAA, and the National Mapping Division of USGS. The database concept was originally formulated during the Specialist Meeting for NCGIA Research Initiative 3 ("Multiple Representations") held in Buffalo, New York in February, 1989.

At that meeting, Frederick Broome (Geography Division, US Census) initiated the idea of disseminating to the general community data culled from several federal agencies. Discussions at the Specialist Meeting included representatives from several federal agencies, including Robert Marx (Census), Richard Berg (Defense Mapping Agency), Charles Schwarz (National Ocean Survey), and Michael Domaratz (USGS). Throughout the following srping and summer, Frederick Broome coordinated much of the agency interaction required for the project.

The discussions for the database focused in part on the need to create a standard of reference upon which to benchmark research on generalization algorithms and database search and query operations. Without a standard data set, it remains difficult to compare the results of various algorithms and to improve our understanding of the nature of digital representations that may change substantially depending upon the resolution at which they are collected, archived, and mapped. A standard data set based on federal products that could be placed in the public domain would provide such a standard. Additionally, this data would provide a useful teaching aid, to demonstrate the use of federal agency data sets, and to explore the transferability of data from one agency format to another.

The NCGIA was a logical agent to represent the academic perspective in the effort, as part of its mandate is to remove impediments to pursuit of research and education in GIS, and to support similar research and educational efforts at other academic institutions through its outreach program. Academic researchers do not always have the contacts or resources to receive and process multiple digital data sets from federal producing agencies. Many academic facilities do not have the resources to extract teaching subsets from larger and complex data sets produced by the federal agencies. The facilities at NCGIA sites are equipped to handle this as a short term service to the academic community, although clearly it is not in the interests of the NCGIA to act as a data clearinghouse on a continuing basis.

Conversely, federal agencies do not maintain small standard data sets to be used or compared with data produced by other agencies. To respond to numerous requests and inquiries for such data sets would take resources from other activities important to the agency's primary mission. The resolution of the community need for a data standard and the finite resources of the agencies mandated a summer internship with labor provided by NCGIA, and funded jointly by US Census and NCGIA. Joseph DeLotto acted as the graduate intern to begin preparation of the data sets, and Barbara Buttenfield served as NCGIA liason for the project.

## 2. Objectives of the Project

Several objectives were defined early on to guide efforts throughout the project, including consideration of federal data producers, of academic data set users, and of dissemination activities of NCGIA. Foremost was the intention that the data sets would significantly aid instructors, researchers, and others interested in problems related to geographic information. An important secondary concern was to minimize the efforts to generate, maintain, and disseminate the data sets to the largest possible community.

For the data producers, only exisiting digital data would be incorporated into the data sets, to minimize efforts required on the part of participating federal agencies. Recipients of the data sets would have at hand a readily available set of geographic data files for incorporation into their GIS curricula. Unpacking software and user documentation would be provided with the data sets to facilitate their use without requiring prior knowledge of binary encoding and data compression. The files would be provided on a cost recovery basis, to maximize dissemination. Data sets would become part of the public domain, to encourage recipients to disseminate files freely to other users, and minimize the long term efforts of NCGIA to serve as a data clearinghouse.

Other objectives were important. Researchers were to have a referent to compare the results of thier research on efficient data structuring and database query, and to benchmark geoprocessing and data generalization tasks. As much of the current research in geoprocessing and generalization involves questions of resolution and scale change, the data should be archived at a variety of scales where feasible. Specific data needs for particular research interests could be served by incorporating data describing a single region produced by several agencies. Integration of these varied data sets became an additional research application tied to the project.

Obviously, research problems unique to specific geographical areas, or unique to a particular agency's data may not be fully served by these data sets. Even in this context, however, the multi-agency data set may provide for testing of early implementations of algorithms or their prototypes, in preparation for later work involving larger data sets obtained directly from a particular agency.

## 3. Data Sets to be Produced

At the outset of the project, a consensus was reached among the participants that study areas for the data sets should be chosen on the basis of three primary criteria. First, the area should be representative of a general physiographic and geographic environment. There should be a variety of digital data available for the area from more than one agency. When feasible, data should be incorporated from sources at more than one map scale.

The representative nature of physical feature types provided from multiple scales and multiple sources would maximize the utility of the data sets for both teaching and research. Accordingly, six regions were selected to capture physiographic diversity and data heterogeneity. The study sites within the regions are determined by overlap of existing digital data coverage by the annotated agencies. In addition to the participating agencies, the list below includes mention of other agencies maintaining spatial data in these regions.

Implementing and testing the production of these data sets was guided by compeltion of an initial data set from the Southeast region, using the Lee County, Florida site (Figure 1). The study site encloses all of Lee County and its surrounding area, and is bounded cleanly by USGS quardrangle limits. The Lee county site offered the most readily available data sets form the greatest number of participating agencies, including Census, USGS, and National Ocean Survey (NOS). Particular source files included in this first data set are listed in Figure 1. It is intended that other agencies may contribute to this and subsequent data sets as they are produced in future years.

## Table 1: Planned Multi-Scale Datasets

| | |
|---|---|
| **Southeast** | **Southern Florida** |
| Census Bureau (TIGER) | |
| Environmental Protection Administration (Pollution site data) | |
| Defense Mapping Agency (Digital elevation data) | |
| Geological Survey (DLG & Mapper data) | |
| National Ocean Service (Chart data) | |
| Soil Conservation Service (Soils data) | |
| **Northeast** | **Chesapeake Bay area** |
| Census Bureau (TIGER) | |
| Environmental Protection Administration (Pollution site data) | |
| Defense Mapping Agency (Digital elevation data) | |
| Geological Survey (DLG & Mapper data) | |
| National Ocean Service (Chart data) | |
| Soil Conservation Service (Soils data) | |
| **Mid-Continent** | **Eastern Nebraska/Southern Iowa** |
| Census Bureau (TIGER) | |
| Environmental Protection Administration (Pollution site data) | |
| Defense Mapping Agency (Digital elevation data) | |
| Geological Survey (DLG & Mapper data) | |
| Soil Conservation Service (Soils data) | |
| **Mountain** | **Colorado/Wyoming** |
| Bureau of Land Management (Federal ownership data) | |
| Environmental Protection Administration (Pollution site data) | |
| Forest Service (Forest data) | |
| Defense Mapping Agency (Digital elevation data) | |
| Geological Survey (DLG & Mapper data) | |
| National Parks Service (Parks data) | |
| Soil Conservation Service (Soils data) | |
| **Southwest** | **Arizona/Utah/New Mexico** |
| Bureau of Indian Affairs (Indian lands data) | |
| Bureau of Land Management (Federal ownership data) | |
| Environmental Protection Administration (Pollution site data) | |
| Forest Service (Forest data) | |
| Defense Mapping Agency (Digital elevation data) | |
| Geological Survey (DLG & Mapper data) | |
| National Parks Service (Parks data) | |
| Soil Conservation Service (Soils data) | |
| **Northwest** | **Washington/Oregon Coast** |
| Bureau of Land Management (Federal ownership data) | |
| Environmental Protection Administration (Pollution site data) | |
| Forest Service (Forest data) | |
| Defense Mapping Agency (Digital elevation data) | |
| Geological Survey (DLG & Mapper data) | |
| National Ocean Service (Chart data) | |
| National Parks Service (Parks data) | |
| Soil Conservation Service (Soils data) | |

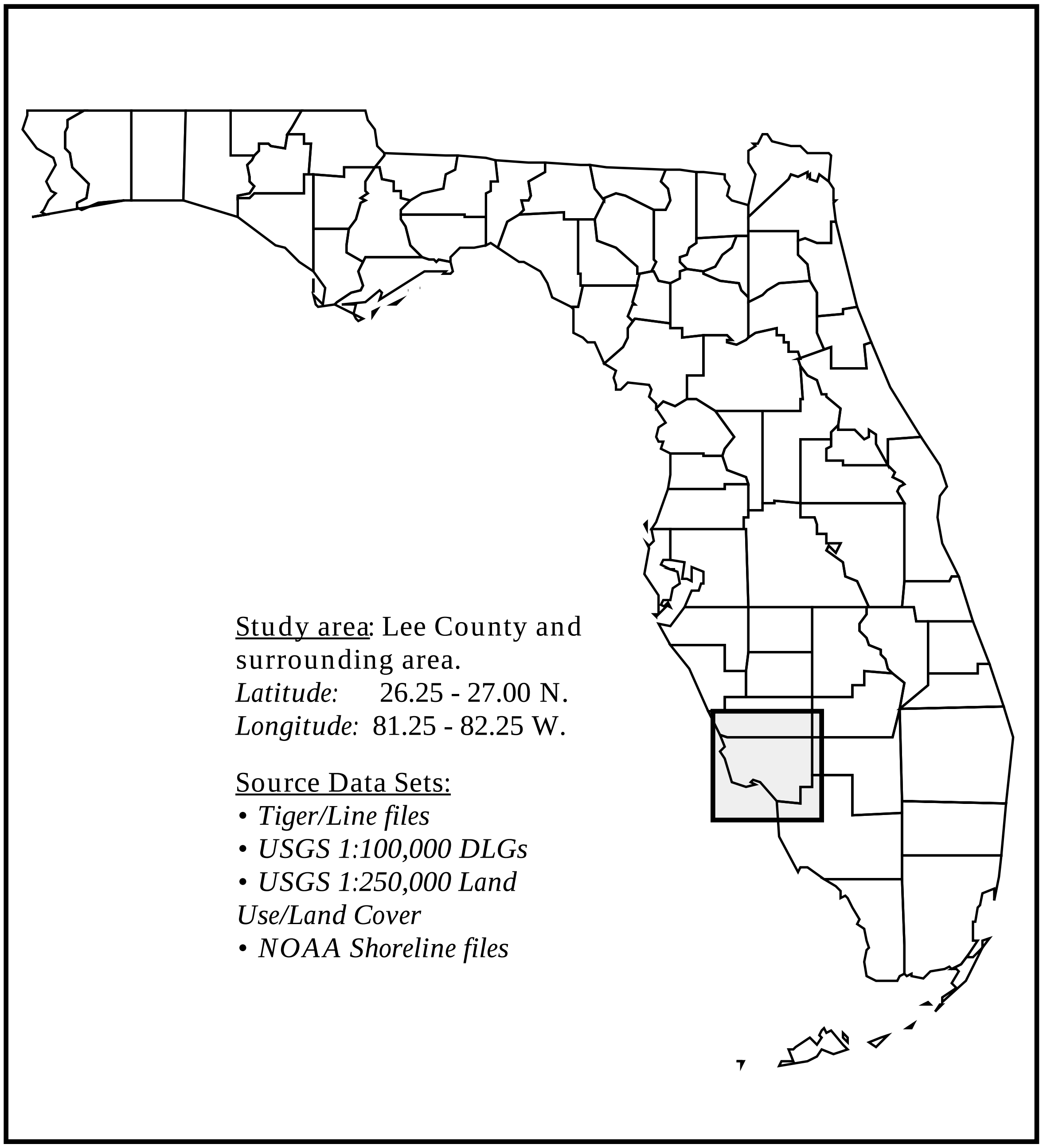

Study area: Lee County and surrounding area.
Latitude: 26.25 - 27.00 N.
Longitude: 81.25 - 82.25 W.
Source Data Sets:
• Tiger/Line files
• USGS 1:100,000 DLGs
• USGS 1:250,000 Land Use/Land Cover
• NOAA Shoreline files

**Figure 1: Multiple Representations Pilot Data Set - Lee County, Florida**

## 4. Current Production and Distribution Plans

The result of this pilot project will be a beta test set to be distributed to NCGIA researchers and a limited number of interested researchers. It is hoped that any bugs in the file format or production software will be weeded out during this test phase, before full-scale production begins on other study areas. Currently, the data is being processed and maintained at NCGIA / Buffalo (see Figure 2), but it is expected that future development and production will be conducted at NCGIA / Santa Barbara, NCGIA / Maine, and at the Census Bureau offices in Suitland, Maryland.

As originally planned, the data sets will be distributed upon request by the NCGIA, at the cost of copying and shipping. Not more than one data set per academic institution or private firm will be provided. Recipients will become customer sites, and agree to distribute the data sets from their own site to other users at that institution (or to other institutions, if they so desire). Further requests to NCGIA from a previous customer site will be directed back to a contact at the customer site. NCGIA will establish the final policy on other dissemination issues.

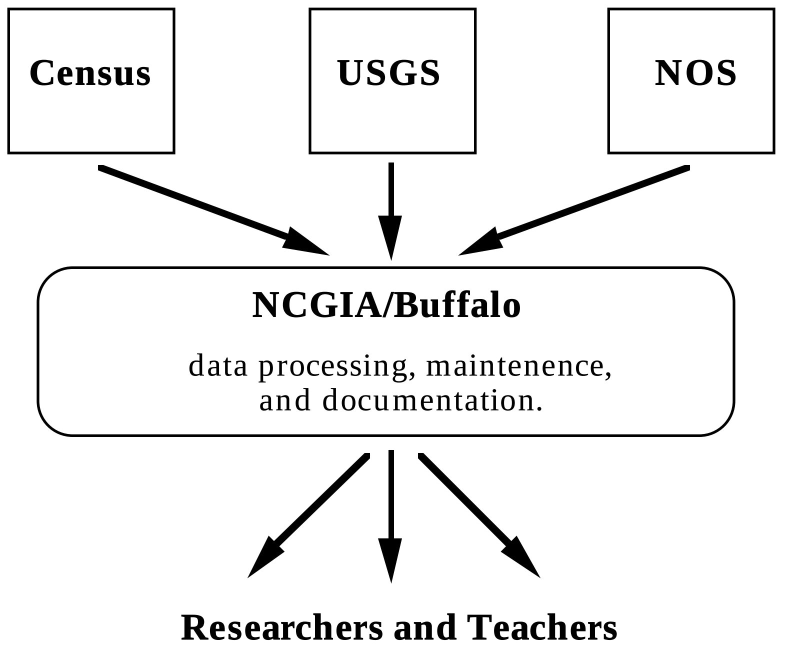


**Figure 2: Current Production Scheme for Data Sets**

The files will be made available in the following formats:

5 1/4 " 1.2MB DOS compatible diskettes
3 1/2" 1.44MB DOS compatible diskettes
3 1/2 Apple compatible diskettes
BitNet (but these are relatively large files even at 2400bpi speed)

Each data set will have a small ASCII instruction file explaining how to unpack and load the data set. The unpacked data sets will be ASCII and not binary even though this is expensive in terms of data space. The user can immediately convert as desired. The documentation will be part of the packed data on the disk(s).

Currently, data being produced on 5 1/4" DOS compatible disks is being compressed with PKware file compression software. Programs PKZIP and PKUNZIP compress the standardized data files an average of 60 - 70%, with some files being compressed as much as 90% (10% of original size). The software, including all optional programs and documentation, is placed on every diskette containing a dataset as a single executable file. By simply typing this filename (PKZ090), the executable will decompress all component files. PKZIP is designated as shareware, although individual users may wish to obtain a site license, which is available from PKware for less than $50.00. PKware may be contacted at:

Phil Katz
PKware, Inc.
7545 North Port Washington Road
Glendale, Wisconsin 53217-3422

Neither NCGIA nor the participating agencies will be expected to respond to request for assistance in use of the files. One of the purposes of the data sets is to remove the need for extensive customer support. However, agencies and NCGIA will establish their own policies in reference to user support.

We intend that the research and teaching community have the opportunity to share their applications and experiences working with this data set. To that end, we propose a 'User Conference' to be held in conjunction with some National Conference, for example the Association of American Geographers National Convention, or the Spring or Fall Meetings of the American Congress on Surveying and Mapping. The 'User Conference' might take the form of a panel, or of a paper session, in which recipients of the data set report on their particular experience. We would like to keep in touch with recipients of the data set to alert them to plans for such a conference; and hope that those who gain copies through the public domain will send their addresses to the NCGIA and ask to be added to the Multi-Agency Data Set Mailing List.

## 5. Technical Discussion

Because the data included in this project comes from multiple agency sources, the first question to be resolved in generating the data set was a choice of format. Agency formats are designed for the purpose to which the digital product is intended, and ranges from navigation (in the case of NOS data) to socioeconomic data analysis (in the case of Census data). A single agency's digital data products may be applied to multiple purposes, and there is a good deal of flexibility in each agency's products to accommodate multiple needs. Still, preprocessing of the source data was necessary to standardize the format for the current project. This will be described in detail below.

Conversion from the various agency formats into a standardized format presented some interesting technical considerations, especially in terms of coordinate transformation and handling. The most troubling computational problems arose from artifacts of the software used to produce the source data files. For instance, Fortran conventions such as 'D' notation in floating point numbers and the suppression of leading zeroes were problematic for transformation programs written in C. These disparities could only be rectified by slow string processing routines to perform in-line replacement of the Fortran 'D's with 'e's to allow floating point reads in C, and filling in of blanks with zeroes to allow string to numeric conversions.

Other complications are more conceptual in nature, and relate to the conception of items archived in the data set. For example, some agencies do not incorporate polygon boundaries, using only point and line features in their files. Others (particularly Census) construct the entire file with topological information. Relating the first type of information to the second can become quite complex in that information in the first file is isolated objects related only implicitly (nearby items will have similar coordinates) while information in the second file requires explicit description of the adjacency and sequencing of the data as it is embedded with other objects in space. The choice of a standard format was in part dictated by these considerations, and a decision was reached to adopt a topologically complete format template, to limit the amount of information loss during processing of data from any agency source.

### 5.1 Rationale for format

To make the data sets more amenable for use with non-commercial software, a single format based on the TIGER/Line structure was used for all of the data sets. The TIGER/Line format was selected as a model because of a number of properties useful to this project. The simplicity of the TIGER record structure made it easy to substitute what attribute information was necessary from each of the agency formats into the existing fixed length records. Because the final data set is to be distributed on 5.25 and 3.5 inch diskettes, the compactness of the 1-cell TIGER structure is especially attractive. Although compactness does limit topological information, it does not compromise geometric representation of features, which is the primary interest of the researchers who will eventually utilize the data.

TIGER/Line coordinates are integer decimal degrees of latitude and longitude with implied 6 decimal precision, which are both easy to read into application programs and easily converted to various projections. Record size, coordinate field specification, and file pointers are consistant for all of the data files produced in this project. However, because of the vastly different attribute information available amoung the different source files used, it was necessary to tailor the attribute fields of the format to each of the individual source files. Although this may not sound particularly 'standard', the data will be used primarily for geometric information, not attribute information.

The next section will discuss the basic file structure of the standardized format, and then individual sections will be devoted to each of the data sources used in this first data set.

## 5.2 Basic file structure of standardized format

Appendix A contains a set of figures showing the logical record structure of the standardized format for TIGER, DLG, GIRAS, and NOS source files. The processing of information into these formats is outlined below in Figure 3.

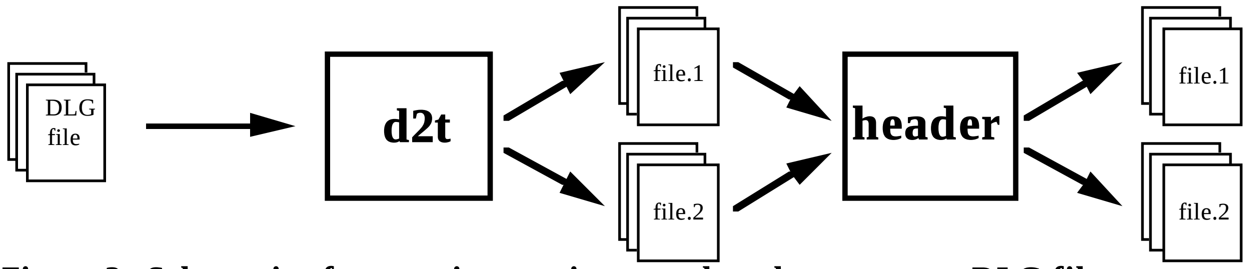


**Figure 3: Schematic of processing routines employed to convert a DLG file to the standard format. This is typical of the processing done to all of the source data files, although some files did need to be clipped to match the study area.**

The modified TIGER/Line structure requires two files, the first containing only type 1 records and the second containing only type 2 records. (Actual TIGER/Line files have these two record types as well as four additional types to carry attribute information such as Census area codes, feature names, and address data.) The type 1 records have attribute information and 'from' and 'to' coordinates, while type 2 records carry only mid-point coordinates for the 1-cell.

In the dataset format, the two record types are linked through a unique record number, with type 1 records being tagged sequentially within the file. Not all type 1 records will have type 2 records associated with them, while others may have multiple type 2 records (up to 999). The only bothersome design flaw with this structure is that type 1 records are not explicitly coded with the number of type 2 records that they possess, so the procedure used to read in the data must check every coordinate for an end of record marker. Details on differences between type 1 and type 2 records are discussed in the sections below.

### 5.2.1 Type 1 Records

The common attribute area for type 1 records is shaded in the record description diagrams shown in the Appendix, and show individual attribute fields as well as leftover space for the source file type. Currently, the leftover space in the type 1 attribute area is left blank, although this space could be utilized in future revisions of the format. All of the datasets will have a three record header area for type 1 files. The standard header is shown below and consists of an identification label, the number of records contained in the file, a source data identifier code, and the extreme coordinates of the file. This particular file contains 39138 type 1 records.

```
**U.S. Census/NCGIA MR standardized data set format**  file type = 1(attribute)
  39138  SOURCE = US Census TIGER file
 78462310 42438007  79312370 43098700
```

Bytes 2 to 6 of the type 1 record will indicate the type of source file. TIGER/Line files will have internal 4-digit census codes for the county that the file covers, DLG's will have 99999, GIRAS land use/land cover data will have 99995, and NOS data will have 99991.

The organization of fields within bytes 16 to 171 of each type 1 record will vary according to the source file from which it was derived. Different record descriptions are necessary for different source files because each source file contains different attribute information. Source file attributes for the data included in the Lee County data set are described for each agency source in turn below.

#### 5.2.1.1 TIGER/Line Data

Figure 1 of Appendix A shows the format of type 1 records for TIGER/Line source files. One of the benefits of using the TIGER/Line structure as a standard is that very little preprocessing is necessary to convert the census-supplied data into the format that is used in this project. In fact, during the original design stage of the datasets, the only alteration required was the addition of a header record. However, in September 1989 the Census Bureau released the 'Precensus' version of the TIGER/Line file, expanding the size of both type 1 and type 2 records. The older format (now designated 'prototype') has been retained in this project because exact compatibility with TIGER, although it was convenient, is not a primary criterion of the structure. Additionally, a number of commercial spatial data handling packages (e.g. ARC/INFO, System 9) have routines that will convert the prototype format into an internal structure, allowing for easy display of the data.

The preprocessing of TIGER/Line files is simply a matter of reorganizing the 'Precensus' format back into the older prototype format. The precise nature of this reorganization can be found in the program pre_proto.c, which is listed in Appendix C. The major modifications made between the formats include:

1) Renumbering of TIGER record numbers from permanent record ID's to 1 through n within each set of related type 1 and type 2 files;

2) Removal of + and - geographic quadrant designations from all coordinates, with west longitude and north latitude being assumed;

3) Elimination of MCD/CCD codes to make room for the expanded place code fields; and

4) Truncation of the leftmost character of American Indian Reservation codes (this may result in loss of reliability for other applications of this data set).

#### 5.2.1.2 USGS Digital Line Graph Data

Figure 2 of Appendix A shows the field arrangement for Digital Line Graph (DLG) data in the standardized format. DLG files contain objects whose geometry is inherent in their code, that is, objects may be categorized as 0-cells instead of points, 1-cells instead of lines, and 2-cells instead of polygon boundaries.

Data codes in these records remain consistent with the major/minor code conventions of USGS. The fixed record size allows for 3 major/minor pairs for the 1-cell, 2 pairs for each of the 2-cells associated with the line, and 1 pair for each of the 0-cells. If more than the allotted number of major/minor pairs exist for a given object, then the retention or deletion of the object is based on the sequence in which the codes are encountered in the DLG. Codes listed first will be retained and later ones will be deleted. Fourteen bytes are leftover in the attribute section of the record for possible future use.

The other major alteration to the data during the reformatting process was the conversion of all coordinates from UTM to geographic (lat/long) coordinate system. Because there was a loss of precision involved in the transformation, the process cannot be inverted to precisely reconstruct the original DLG data.

### 5.2.1.3 NOS Shoreline Data

NOS shoreline data for the Florida data set was made available to this project as a single file in NOS Nautical Charting Division Standard Digital Data Exchange Format (NCD SDDEF) Version 1 of April 1, 1985. Specific record and field descriptions for this format are given in Appendix B. More generally, the format consists of 80 character fixed-length records, each representing a 0-cell. The next to last character of each record indicates whether the 0-cell is an endpoint of a line segment, a midpoint of a line segment, or a point feature. No polygon information is represented in the NOS data.

Conversion of NOS data to TIGER-like format is fairly straightforward, and the modified record description is shown in figure 3 of appendix A. The code for performing this conversion has been included in appendix C (nos_tiger.c). All of the fields in the attribute area of the type 1 record shown in figure n have been placed directly from the NOS file without modification, and the field names match the original NOS descriptions.

A slight size adjustment has been made for the fields 'line type' and 'geographic quadrant', which have been made two characters long instead of one to allow for the possibility of a -1 value, which indicates that the field was empty in the original file. The only NOS field to be excluded from the new data record is 'curve endpoints'. This information is now implicit in the new structure, where all records are 1-cells unless the from coordinates are the same as the to coordinates, indicating a 0-cell.

### 5.2.1.4 USGS Land use/Land cover Data

Figure 4 of Appendix A shows the format for USGS land use/land cover data. Although it is not particularly space efficient, it is far more user-accessible than in its former life as a GIRAS file. There are a number of difficulties with this data, including use of 'file sections', and a lack of definitions for attribute codes. Users of this particular source data may wish to consult GIRAS documentation available from USGS.

Problems with 'file sections' were encountered relating to the design limitations in the older Geographic Information Analysis and Retrieval System (GIRAS) format. GIRAS format conventionnally split map files into 'sections' when the volume of data exceeded a certain threshold imposed by limitations on computer memory. While those limiatations are now obsolete, the formatting convention has been maintained in the data format. A single data file can contain a number of these sections, each with its own numbering (1 to n) of map features. When translated to the standardized format these section numbers are included, but because of feature labels are non-unique they must be processed differently by user applications.

Additionally, GIRAS formats use a system of local coordinates which utilize the nearest 100,000 meter UTM grid intersection to the south and west of a map section as the local origin. Unfortunately, this intersection is not specified anywhere in the file header, and must therefore be computed. In order to convert the local coordinates to lat-long, latitude and longitude values of the map corners provided in the file header were put through a forward UTM projection program to establish what the local coordinate origin had been. This origin was then added to each of the local coordinates, and an inverse transformation was performed to get back geographic coordinates needed for the TIGER/Line format. Some variation in precision was introduced in the course of the two transformations due to machine rounding during computations.

## 5.2.2 Type 2 Records

Figure 5 of Appendix A shows the type 2 record layout that is used for all of the type 1 records, regardless of source file. Type 2 records are associated with the type 1 record through the record number (located in bytes 7 through 14). These records will hold up to 10 Latitude/Longitude coordinate pairs. Line segments containing than 10 pairs will have multiple type 2 records, which are ordered within the type 2 file through the use the 3 digit 'record sequence' field. If any type 2 record has less than 10 points, it is padded with zeroes up to the 187 byte record length.

Header information for type 2 records follows the format showed in the section on type 1 records. However, whereas the organization of fields within bytes 16 to 171 of each type 1 record varies according to its source file, as described in the previous sections, type 2 records will remain the same for all source files.

## 6. Project status

The Lee County pilot project is now complete, and ready for dissemination into the research and teaching community. Agency cooperation for production of further data sets continues to be very high. In the coming summer, an intern from NCGIA-Santa Barbara will continue working on the project, and will take on responsibilities for disseminating the Lee County data. We hope that testing and use of the data will begin in the summer months and coming academic year. A shortened version of this document will be submited to a professional journal, to ensure the widest possible visibility for the data set.

## 7. Documentation

On the next page is a copy of the actual documentation included with each data set as it will be distributed. The documentation will be provided in both electronic and hardcopy form with the actual data, to assist users in unpacking the data set and understanding its components.

The data files included on the diskettes are the product of a joint effort of the National Center for Geographic Information and Analysis at Buffalo, and the U.S. Bureau of the Census. The intent of the project is to provide multiple coverages of a single geographic area with data derived from a number of different sources, as well as at a number of scales. It is hoped that researchers in spatial data handling can use this data to explore various issues of multiple representations of spatial data, and as a benchmark dataset for developing and testing algorithms. The data should also serve as an aid to teaching and course exercises in GIS and digital cartography. This data set covers Lee County, Florida, and environs. It is the pilot data set, and tests the feasibility of specific data formats and compression software. Included in this release are files derived from USGS Digital Line Graph data, USGS Land Use/Land Cover data, Census TIGER files, and NOS Vector Shoreline files. The geographic window is coincident with boundaries for USGS quadrangles, and cover the Lee County boundaries completely. Specific descriptions of the file formats are given in NCGIA Report 90-4, available from NCGIA, Department of Geography, University of California, Santa Barbara, CA 93106.

Documentation for the file compression program PKZIP has been included with a copy of the compression software on each data diskette. The directory of each diskette will resemble the following (generic) directory:

```
README
PKZ092.EXE
XXXXX.ZIP
XXXXX.ZIP
XXXXX.ZIP
```

The README file contains an electronic version of this text. To see it on a CRT, simply use the DOS 'type' command or bring it into any text editor on a Macintosh. The PKZ092.EXE file contains the software and documentation for the PKWARE compression routines. You should load this onto your hard disk (in its own directory or folder), and type PKZ092 to run the program to unpack 14 PKWARE programs. The only one you will need is PKUNZIP.EXE; the rest have been provided as a matter of shareware etiquette.

To unpack any particular file, type:

```
PKUNZIP   XXXXXX.ZIP
```

where XXXXX.ZIP is the name of a compressed file on the diskette. Expect an expansion of 8 to 9 times the compressed file sizes when you unpack this data set. Make sure there will be enough space on your hard disk before you run the program, otherwise the unpacking will not execute successfully.

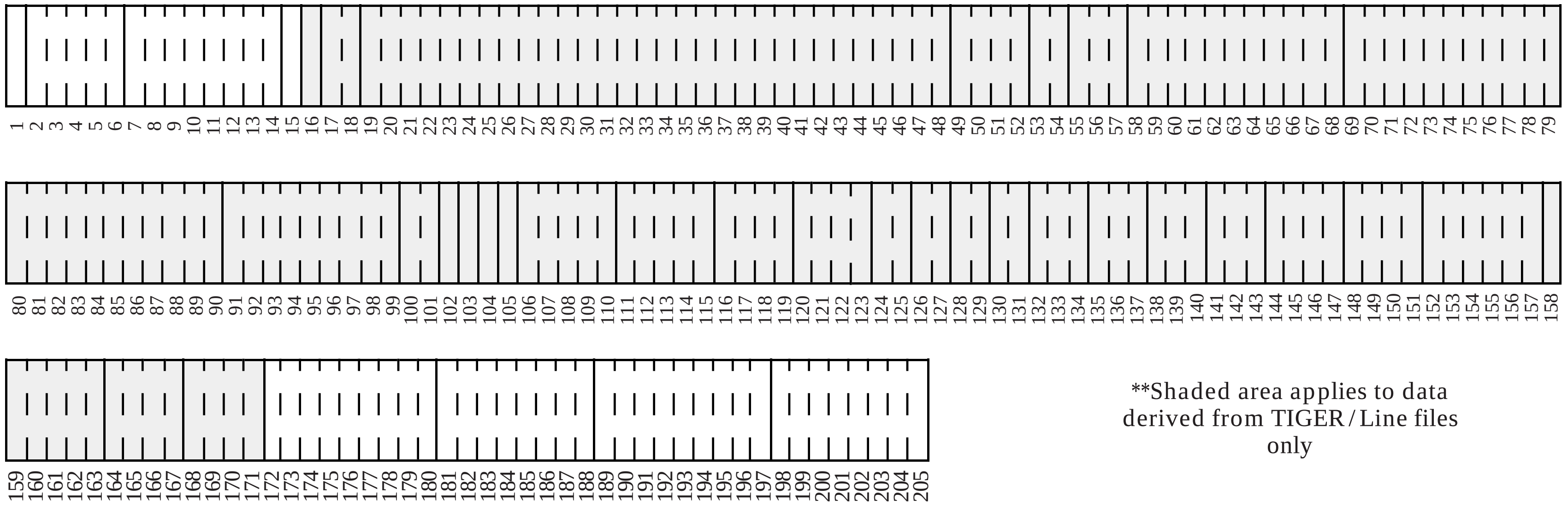
***NOTE - In the Lee county data set, the type 1 file for the Lee county Tiger file was too large to fit on a single diskette. The file was split into two parts (head_lee.1 and tail_lee.1) and placed on separate diskettes. Before using these files, they should be put back together to form a single file preferably named lee.1 (as the type 2 file is named lee.2). After unzipping head_lee.1 and tail_lee.1, the following DOS command should be executed:

```
copy head_lee.1 + tail_lee.1 lee.1
```

which will concatenate the files into file lee.1. Files head_lee.1 and tail_lee.1 may then be deleted.

# Appendix A
Data Description Diagrams

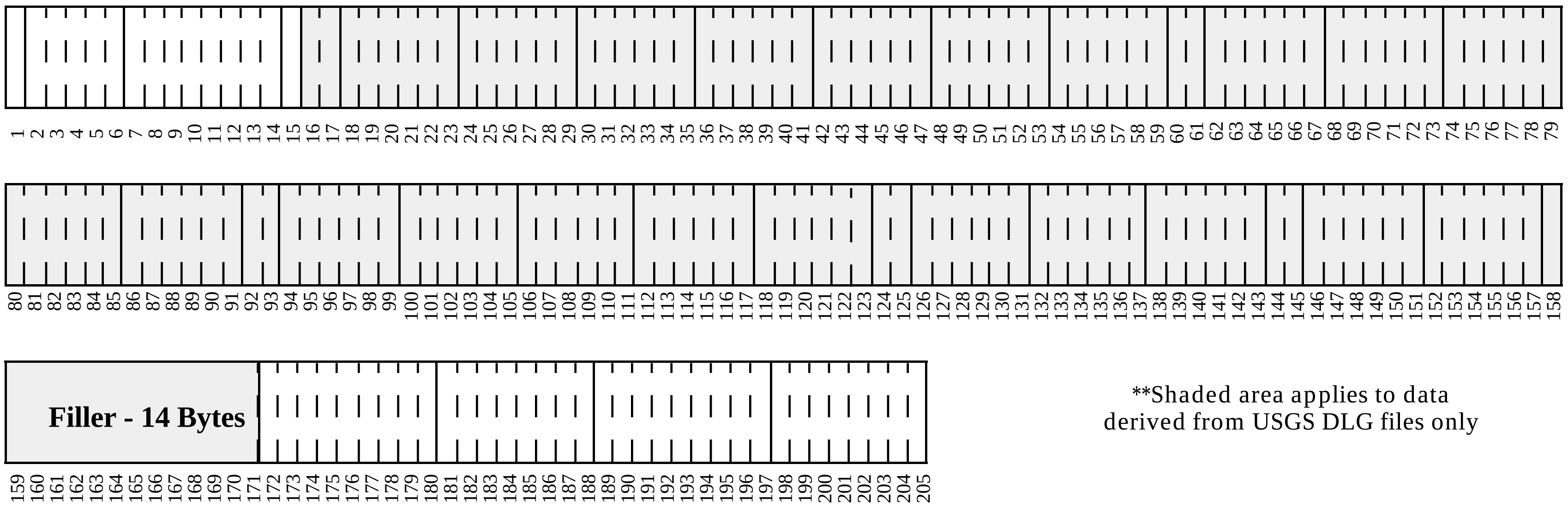
## Figure 1: TIGER/Line Source Data Record Description Diagram - record type I (attribute)



**Shaded area applies to data derived from TIGER / Line files only

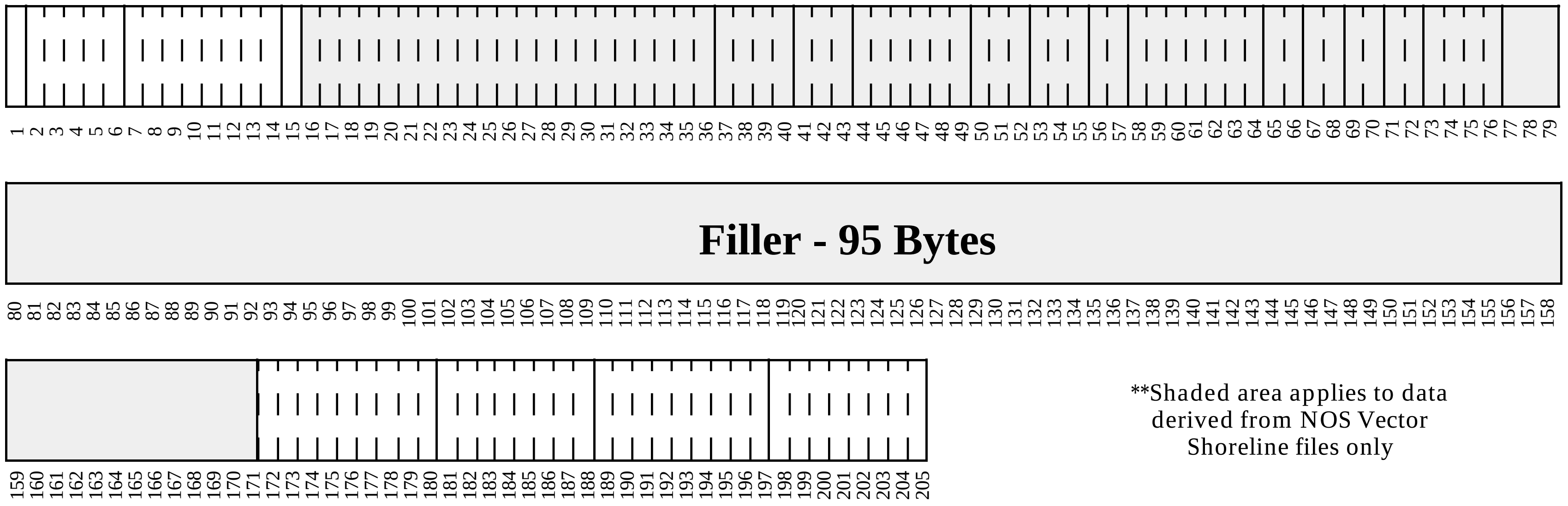
| bytes | | bytes | | bytes | | bytes | |
|---|---|---|---|---|---|---|---|
| 1 | record type (value 1) | 58-68 | left from address | 116-119 | American Indian res. left | 144-147 | place code left |
| 2-6 | file code (FIPS code) | 69-79 | left to address | 120-123 | American Indian res. right | 148-151 | place code right |
| 7-14 | record number | 80-90 | right from address | 124-125 | Alaska native corp. left | 152-157 | Census tract/BNA left |
| 15 | single side code | 91-101 | right to address | 126-127 | Alaska native corp. right | 158-163 | Census tract/BNA right |
| 16 | source code | 102 | left from flag | 128-129 | FIPS state code left | 164-167 | tabulation block left |
| 17-18 | feature direction (prefix) | 103 | left to flag | 130-131 | FIPS state code right | 168-171 | tabulation block right |
| 19-48 | feature name | 104 | right from flag | 132-134 | FIPS county code left | 172-180 | from longitude |
| 49-52 | feature type | 105 | right to flag | 135-137 | FIPS county code right | 181-188 | from latitude |
| 53-54 | feature direction (suffix) | 106-110 | ZIP code left | 138-140 | MCD/CCD code left | 189-197 | to longitude |
| 55-57 | census feature class code | 111-115 | ZIP code right | 141-143 | MCD/CCD code right | 198-205 | to latitude |

## **Figure 2:** USGS DLG Source Data Record Description Diagram - record type I (attribute)


Filler - 14 Bytes
**Shaded area applies to data derived from USGS DLG files only

| bytes | | bytes | | bytes | |
|---|---|---|---|---|---|
| 1 | record type (value 1) | 60-61 | # of area right major / minor codes | 124-125 | # of from node major / minor codes |
| 2-6 | file code (value 99999) | 62-67 | area right major code 1 | 126-131 | from node major code 1 |
| 7-14 | record number | 68-73 | area right minor code 1 | 132-137 | from node minor code 1 |
| 15 | single side code | 74-79 | area right major code 2 | 138-143 | to node id - number |
| 16-17 | # of major / minor codes for line | 80-85 | area right minor code 2 | 144-145 | # of to node major / minor codes |
| 18-23 | line major code 1 | 86-91 | area left id - number | 146-151 | to node major code 1 |
| 24-29 | line minor code 1 | 92-93 | # of area left major / minor codes | 152-157 | to node minor code 1 |
| 30-35 | line major code 2 | 94-99 | area left major code 1 | 158-171 | filler |
| 36-41 | line minor code 2 | 100-105 | area left minor code 1 | 172-180 | from longitude |
| 42-47 | line major code 3 | 106-111 | area left major code 2 | 181-188 | from latitude |
| 48-53 | line minor code 3 | 112-117 | area left minor code 2 | 189-197 | to longitude |
| 54-59 | area right id - number | 118-123 | from node id - number | 198-205 | to latitude |

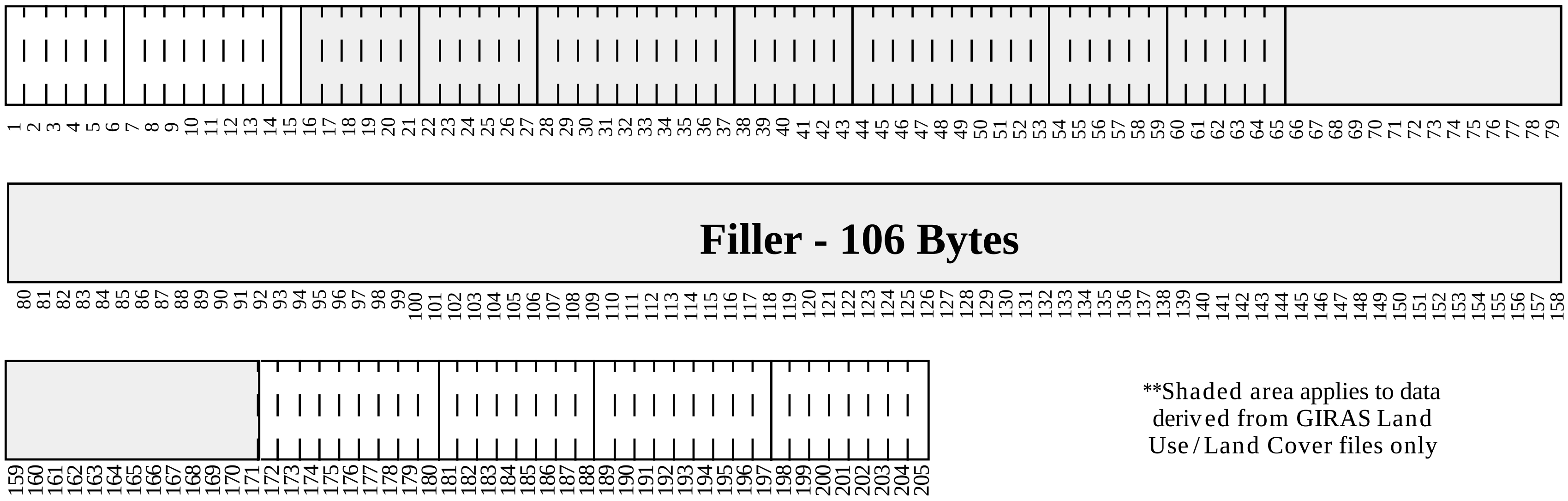
# Figure 3: NOS Vector Shoreline Source Data Record Description Diagram - record type I (attribute)



**Shaded area applies to data derived from NOS Vector Shoreline files only

| bytes | | bytes | | bytes | |
|---|---|---|---|---|---|
| 1 | record type (value 1) | 50-52 | day of location | 73-76 | cartographic feature code |
| 2-6 | file code (value 99995) | 53-55 | year of location | | |
| 7-14 | record number (0 padded) | 56-57 | quality code | | |
| 15 | single side code (unused) | 58-64 | depth / height | | |
| 16-36 | name of feature | 65-66 | line type | | |
| 37-40 | scale | 67-68 | geographic quadrant | | |
| 41-43 | source document code | 69-70 | horizontal datum code | | |
| 44-49 | source document registry number | 71-72 | vertical datum code | | |

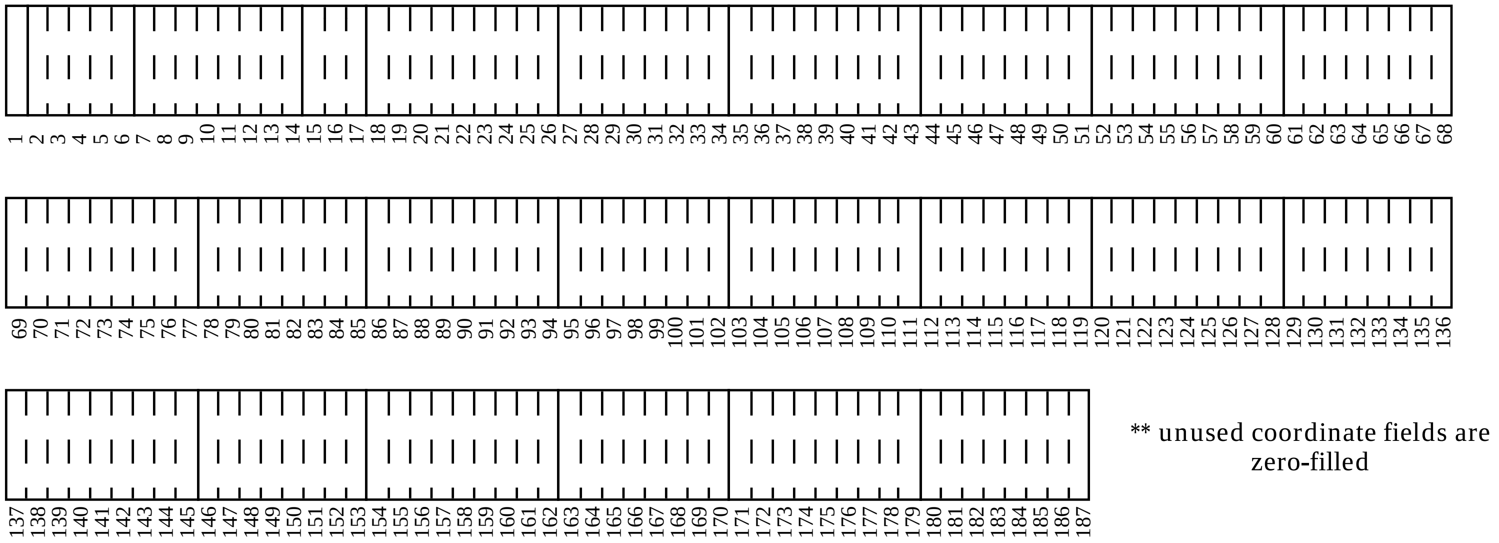
## Figure 4: USGS Land Use/Land Cover Source Data Record Description Diagram - record type I (attribute)



**Shaded area applies to data derived from GIRAS Land Use/Land Cover files only

| bytes | | bytes | |
|---|---|---|---|
| 1 | record type (value 1) | 44-53 | polygon right attribute code |
| 2-6 | file code (value 99995) | 54-59 | start node |
| 7-14 | record number (0 padded) | 60-65 | end node |
| 15 | single side code (unused) | 66-171 | filler |
| 16-21 | section number of record | 172-180 | from longitude |
| 22-27 | polygon left ID | 181-188 | from latitude |
| 28-37 | polygon left attribute code | 189-197 | to longitude |
| 38-43 | polygon right ID | 198-205 | to latitude |

## Figure 5: NCGIA Standard Data Record Description Diagram - record type II (coordinates)



| bytes |  | bytes |  | bytes |  |
|---|---|---|---|---|---|
| 1 | record type (value 2) | 52-60 | longitude # 3 | 120-128 | longitude # 7 |
| 2-6 | file code | 61-68 | latitude # 3 | 129-136 | latitude # 7 |
| 7-14 | record number (0 padded) | 69-77 | longitude # 4 | 137-145 | longitude # 8 |
| 15-17 | record sequence | 78-85 | latitude # 4 | 146-153 | latitude # 8 |
| 18-26 | longitude # 1 | 86-94 | longitude # 5 | 154-162 | longitude # 9 |
| 27-34 | latitude # 1 | 95-102 | latitude # 5 | 163-170 | latitude # 9 |
| 35-43 | longitude # 2 | 103-111 | longitude # 6 | 171-179 | longitude # 10 |
| 44-51 | latitude # 2 | 112-119 | latitude # 6 | 180-187 | latitude # 10 |

# Appendix B
# Directory of Data Files for Lee County Data Set

**Disk #1**

| compressed file | uncompressed files |
|---|---|
| c_harb1.zip | zc2hyf4.1 |
| | zc2mtf4.1 |
| | zc2rdf4.1 |
| | zc2rrf4.1 |
| | zc2hyf4.2 |
| | zc2mtf4.2 |
| | zc2rdf4.2 |
| | zc2rrf4.2 |

| | |
|---|---|
| geographic extent: | 82.0000W - 82.2500W |
| | 26.7500N - 27.0000N |
| uncompressed size: | 2.89 mbytes |
| source file: | USGS DLG |

| compressed file | uncompressed files |
|---|---|
| c_harb2.zip | zc2hyf8.1 |
| | zc2mtf8.1 |
| | zc2rdf8.1 |
| | zc2rrf8.1 |
| | zc2hyf8.2 |
| | zc2mtf8.2 |
| | zc2rdf8.2 |
| | zc2rrf8.2 |

| | |
|---|---|
| geographic extent: | 82.0000W - 82.2500W |
| | 26.5000N - 26.7500N |
| uncompressed size: | 1.48 mbytes |
| source file: | USGS DLG |

| compressed file | uncompressed files |
|---|---|
| ftmyers1.zip | wp1hyf1.1 |
| | wp1mtf1.1 |
| | wp1rdf1.1 |
| | wp1rrf1.1 |
| | wp1hyf1.2 |
| | wp1mtf1.2 |
| | wp1rdf1.2 |
| | wp1rrf1.2 |

| | |
|---|---|
| geographic extent: | 81.7500W - 82.0000W |
| | 26.7500N - 27.0000N |
| uncompressed size: | 0.98 mbytes |
| source file: | USGS DLG |

| compressed file | uncompressed files |
|---|---|
| ftmyers2.zip | wp1hyf2.1 |
| | wp1mtf2.1 |
| | wp1rdf2.1 |
| | wp1rrf2.1 |
| | wp1hyf2.2 |
| | wp1mtf2.2 |
| | wp1rdf2.2 |
| | wp1rrf2.2 |

| | |
|---|---|
| geographic extent: | 81.5000W - 81.7500W |
| | 26.7500N - 27.0000N |
| uncompressed size: | 0.59 mbytes |
| source file: | USGS DLG |

## **Disk #2**

| compressed file | uncompressed files |
|---|---|
| ftmyers3.zip | wp1hyf5.1 |
| | wp1mtf5.1 |
| | wp1rdf5.1 |
| | wp1rrf5.1 |
| | wp1hyf5.2 |
| | wp1mtf5.2 |
| | wp1rdf5.2 |
| | wp1rrf5.2 |

| | |
|---|---|
| geographic extent: | 81.7500W - 82.0000W |
| | 26.5000N - 26.7500N |
| uncompressed size: | 3.38 mbytes |
| source file: | USGS DLG |

| compressed file | uncompressed files |
|---|---|
| ftmyers4.zip | wp1hyf6.1 |
| | wp1mtf6.1 |
| | wp1rdf6.1 |
| | wp1rrf6.1 |
| | wp1hyf6.2 |
| | wp1mtf6.2 |
| | wp1rdf6.2 |
| | wp1rrf6.2 |

| | |
|---|---|
| geographic extent: | 81.5000W - 81.7500W |
| | 26.5000N - 26.7500N |
| uncompressed size: | 1.48 mbytes |
| source file: | USGS DLG |

| compressed file | uncompressed files |
|---|---|
| naples1.zip | wp3hyf1.1 |
| | wp3mtf1.1 |
| | wp3rdf1.1 |
| | wp3rrf1.1 |
| | wp3hyf1.2 |
| | wp3mtf1.2 |
| | wp3rdf1.2 |
| | wp3rrf1.2 |

| | |
|---|---|
| geographic extent: | 81.7500W - 82.0000W |
| | 26.2500N - 26.5000N |
| uncompressed size: | 1.14 mbytes |
| source file: | USGS DLG |

| compressed file | uncompressed files |
|---|---|
| naples2.zip | wp3hyf2.1 |
| | wp3mtf2.1 |
| | wp3rdf2.1 |
| | wp3rrf2.1 |
| | wp3hyf2.2 |
| | wp3mtf2.2 |
| | wp3rdf2.2 |
| | wp3rrf2.2 |

| | |
|---|---|
| geographic extent: | 81.5000W - 81.7500W |
| | 26.2500N - 26.5000N |
| uncompressed size: | 0.92 mbytes |
| source file: | USGS DLG |

| compressed file | uncompressed files |
|---|---|
| sanibel1.zip | wp3hyf2.1 |
| | wp3mtf2.1 |
| | wp3rdf2.1 |
| | wp3rrf2.1 |
| | wp3hyf2.2 |
| | wp3mtf2.2 |
| | wp3rdf2.2 |
| | wp3rrf2.2 |

| | |
|---|---|
| geographic extent: | 81.5000W - 81.7500W |
| | 26.2500N - 26.5000N |
| uncompressed size: | 0.92 mbytes |
| source file: | USGS DLG |

**Disk #3**

| compressed file | uncompressed files |
|---|---|
| wpb250.zip | wpbcen.1 |
| | wpbhyd.1 |
| | wpbfed.1 |
| | wpbland.1 |
| | wpbpol.1 |
| | wpbstate.1 |
| | wpbcen.2 |
| | wpbhyd.2 |
| | wpbfed.2 |
| | wpbland.2 |
| | wpbpol.2 |
| | wpbstate.2 |

| | |
|---|---|
| geographic extent: | 81.5000W - 82.0000W |
| | 26.2500N - 27.0000N |
| uncompressed size: | 2.70 mbytes |
| source file: | USGS Land Use/Land Cover |

| compressed file | uncompressed files |
|---|---|
| nosdata.zip | nos.1 |
| | nos.2 |

| | |
|---|---|
| geographic extent: | 81.5000W - 82.2500W |
| | 26.2500N - 27.0000N |
| uncompressed size: | 3.63 mbytes |
| source file: | NOS Vector Shoreline |

| compressed file | uncompressed files |
|---|---|
| Hendry.zip | Hendry.1 |
| | Hendry.2 |

| | |
|---|---|
| geographic extent: | 81.5000W - 81.5665W |
| | 26.5129N - 26.7693N |
| uncompressed size: | 0.18 mbytes |
| source file: | Tiger/Line |

**Disk #4**

| compressed file | uncompressed files |
|---|---|
| head_lee.zip | head_lee.1 |

geographic extent: 81.5624W - 82.2500W
26.3159N - 26.7865N
uncompressed size: 6.21 mbytes
source file: Tiger/Line

**Disk #5**

| compressed file | uncompressed files |
|---|---|
| tail_lee.zip | tail_lee.1 |

geographic extent: 81.5624W - 82.2500W
26.3159N - 26.7865N
uncompressed size: 6.07 mbytes
source file: Tiger/Line

| compressed file | uncompressed files |
|---|---|
| Glades.zip | Glades.1 |
| | Glades.2 |

geographic extent: 81.2500W - 81.5663W
26.7682N - 27.0000N
uncompressed size: 0.65 mbytes
source file: Tiger/Line

**Disk #6**

| compressed file | uncompressed files |
|---|---|
| lee.zip | lee.2 |

geographic extent: 81.5624W - 82.2500W
26.3159N - 26.7865N
uncompressed size: 2.44 mbytes
source file: Tiger/Line

| compressed file | uncompressed files |
|---|---|
| Collier.zip | Collier.1 |
| | Collier.2 |

geographic extent: 81.2500W - 81.8820W
26.2500N - 26.5167N
uncompressed size: 1.94 mbytes
source file: Tiger/Line

**Disk #7**

| compressed file | uncompressed files |
|---|---|
| Charlott.zip | Charlott.1 |
| | Charlott.2 |

| | |
|---|---|
| geographic extent: | 81.5629W - 82.2500W |
| | 26.7683N - 27.0000N |
| uncompressed size: | 6.08 mbytes |
| source file: | Tiger/Line |

# Appendix C
# Processing routines and component programs

## d2t

Description: creates standardized file types (tiger/line) from a 1:100,000 USGS DLG file (optional format).
Input: will prompt for DLG file name and output file names for the attribute and coordinates files.
Output: creates or overwrites 2 files.
Component source files: dlg_to_tiger.c, utm_ll.c, joe.h, dlghead.h.

## clip

Description: clips rectangular window from standardized file type -- does not alter attribute information.
Input: will prompt for names of the input and output attribute and coordinates files, and the 4 min-max coordinates that define the clipping window.
Output: creates or overwrites 2 files.
Component source files: clip.c, cohen_suth.c, joe.h, clip.h.

## giras2t

Description: creates standardized file types (tiger/line) from GIRAS format data (USGS Land Use/Land Cover).
Input: will prompt for the GIRAS file name and the names of the output attribute and coordinates files.
Output: creates or overwrites 2 files.
Component source files: giras_read.c, utm_ll.c, ll_utm.c, joe.h, giras.h.

## header

Description: attaches header to standardized file formats.
Input: will prompt for names of the input attribute and coordinates files, which will also be used as the output file names.
Output: overwrites 2 files.
Component source files: header.c, joe.h, clip.h.

**** cohen_suth.c, utm_ll.c, ll_utm.c, alber_fwd.c may be compiled and run independently as test programs.**

```
/*************************************************************************
******/
```

**cohen_suth.c**

```
/*      ----->function cohen_suth.c is an implementation of the
              Cohen-Sutherland clipping algorithm.  This version uses
              bit manipulations on short integers to produce the 4 bit
              outcodes as described in the original algorithm.  Line
              intersections are found through a recursive midpoint
              subdivision process.
```

```
                parameters
            ----------
                x1,y1,x2,y2 - (integers) coordinates of line endpoints.
                max_x,min_x,max_y,min_y - (integers) maximum and minimum
                        extent of the clipping window.
                x,y - coordinates of intersection point.

                return values
                -------------
                0 = line is trivially outside the window
                1 = line is trivially inside the window
                2 = intersection found and returned as x,y
 */

#include <stdio.h>


int cohen_suth(x1,y1,x2,y2,max_x,min_x,max_y,min_y,x,y)
int     x1,y1,x2,y2,max_x,min_x,max_y,min_y,*x,*y;

{
short   outcode1,outcode2,accept,reject;
short   reject_check(),accept_check(),outcodes();

accept = 0;
reject = 0;
outcode1 = 0;
outcode1 = outcodes(x1,y1,max_x,min_x,max_y,min_y);
outcode2 = 0;
outcode2 = outcodes(x2,y2,max_x,min_x,max_y,min_y);
reject = reject_check(outcode1,outcode2);
if (reject)
        return(0);
else
        {
        accept = accept_check(outcode1,outcode2);
        if (accept)
                {
                *x = x2;
                *y = y2;
                return (1);
                }
        else
                {
                if (!outcode1)
                        swap_pts(&x1,&y1,&x2,&y2,&outcode1,&outcode2);
                *x = (x2 + x1)/2;
                *y = (y2 + y1)/2;
                if(((*x == x1)&&(*y == y1))||((*x == x2)&&(*y == y2)))
                        {
                        accept = 1;
                        *x = x2;
                        *y = y2;
                        return(2);
                        }
        if (!accept)
                accept=cohen_suth(x1,y1,*x,*y,max_x,min_x,max_y,min_y,x,y);
                if (!accept)
                accept=cohen_suth(*x,*y,x2,y2,max_x,min_x,max_y,min_y,x,y);
                }
        }
return(accept);
}


swap_pts(x1,y1,x2,y2,outcode1,outcode2)
int     *x1,*y1,*x2,*y2;
short   *outcode1,*outcode2;

{
int     temp;

temp = *x1;
*x1 = *x2;
*x2 = temp;
temp = *y1;
*y1 = *y2;
*y2 = temp;
temp = *outcode1;
*outcode1 = *outcode2;
*outcode2 = temp;
}


short accept_check(code1,code2)
short           code1,code2;

{
if (code1 == 0 && code2 == 0)
        return(1);
else
        return(0);
}


short reject_check(code1,code2)
short           code1,code2;

{
short           result;

result = (code1 & code2);
if (!result)
        return(0);
else
        return(1);
}
```

```
short outcodes(x,y,max_x,min_x,max_y,min_y)
int        x,y,max_x,min_x,max_y,min_y;

{
short    outcode;

outcode = 0;
if ((max_y - y) < 0)
           outcode = outcode | 8;
if ((y - min_y) < 0)
           outcode = outcode | 4;
if ((max_x - x) < 0)
           outcode = outcode | 2;
if ((x - min_x) < 0)
           outcode = outcode | 1;
return(outcode);
}


#ifdef TEST          /*   -----> this is a test harness for cohen_suth.c  */
main()
{
int        x1,y1,x2,y2,min_x,max_x,min_y,max_y,x,y;
int        test,again;

test = 0;
printf("Welcome to test harness of utm_ll.c\n");
printf("\nEnter bounding coordinates for window to be clipped\n");
get_xy(&max_x,&max_y,&min_x,&min_y,1);
again = 1;
while(again)
           {
           printf("\nEnter start and end points of the line to be clipped\n");
           get_xy(&x1,&y1,&x2,&y2,2);
           test = cohen_suth(x1,y1,x2,y2,max_x,min_x,max_y,min_y,&x,&y);
           if (test == 0)
                      printf("line is trivially outside of the window\n");
           else if (test == 1)
                      printf("line is trivially inside of the window\n");
           else if (test == 2)
                      {
                      printf("x = %d\n",x);
                      printf("y = %d\n",y);
                      }
           printf("\nenter 0 to stop: ");
           scanf("%d",&again);
           }
}
#endif
```

```
/*************************************************************************
*******/


get_xy(x1,y1,x2,y2,ind)
int       *x1,*y1,*x2,*y2;
short     ind;

{
if (ind == 1)
          {
          printf("\tmaximum x:");
          scanf("%d",x1);
          printf("\tmaximum y:");
          scanf("%d",y1);
          printf("\tminimum x:");
          scanf("%d",x2);
          printf("\tminimum y:");
          scanf("%d",y2);
          }
if (ind == 2)
          {
          printf("\tx1:");
          scanf("%d",x1);
          printf("\ty1:");
          scanf("%d",y1);
          printf("\tx2:");
          scanf("%d",x2);
          printf("\ty2:");
          scanf("%d",y2);
          }
}
```

```
/***************************************************************************
*******/
```

**ll_utm.c**

```
/*	----->function ll_utm:  accepts parameters y (longitude) and x
 *		(latitude), which must be integer values with implied
 *		three decimal precision. They are assumed to be in decimal
 *		degree format.  It returns y and x as UTM coordinates with
 *		implied two decimal precision (integers), as well as
 *		identifying the appropriate zone.
 *		**NOTE: this function has been written for areas in N. America,
 *		and will treat signed and unsigned coordinates as if they
 *		are in the north and west hemispheres REGARDLESS of the sign.
 */

#include <math.h>
#include <stdio.h>
#include "joe.h"
#define	alpha	6378206.40	/*   ----->set parameters for Clarke	*/
#define	e_square 0.00676866	/*	1866 ellipsoid		*/
#define	phi_zero 0
#define	k_zero	0.9996
#define	rads	PI/180		/*   ----->factor for radian conversions */



ll_utm(y,x,zone)
int	*y,*x,*zone;

{
double	ep_square,lam_zero,mo,m,cosphi,
	phi,lambda,c,t,n,a,newx,newy;
int	yesno,z_check;


/*	----->initialize zone to zero, and cast the latitude and longitude
 *		into floating point numbers with the correct sign.
 */
	*zone = 0;
	phi = (float) ABS(*y)/10000000;
	lambda = (float) ANTI_ABS(*x)/10000000;

/*	----->convert both phi and lambda to radians for now
 */
	phi *= rads;
	lambda *= rads;

/*	----->establish UTM zone.  Any points falling on dividing meridians
 *		will cause an error message.  Cross zone computations should
 *		use altered version of this function.
 */
```

```
z_check = ABS(*x);
if (z_check > 600000000 && z_check < 660000000)
        *zone = 20;
else if (z_check > 660000000 && z_check < 720000000)
        *zone = 19;
else if (z_check > 720000000 && z_check < 780000000)
        *zone = 18;
else if (z_check > 780000000 && z_check < 840000000)
        *zone = 17;
else if (z_check > 840000000 && z_check < 900000000)
        *zone = 16;
else if (z_check > 900000000 && z_check < 960000000)
        *zone = 15;
else if (z_check > 960000000 && z_check < 1020000000)
        *zone = 14;
else
        {
        printf("\n\7** WARNING - ambiguous zone designation for coordinate");
        printf(" pair:\n");
        printf("\tx = %d\n",*x);
        printf("\ty = %d\n",*y);
        printf("do you want to manually enter zone designation(1/0):");
        scanf("%d",&yesno);
        if (yesno)
                {
                printf("enter UTM zone:");
                scanf("%d",zone);
                }
        else
                {
                printf("returning control to _main with null value\n");
                return(0);
                }
        }

/*
 */

switch (*zone) {
        case 14:
                lam_zero = -99*rads;
        break;
        case 15:
                lam_zero = -93*rads;
        break;
        case 16:
                lam_zero = -87*rads;
        break;
        case 17:
                lam_zero = -81*rads;
        break;
        case 18:
                lam_zero = -75*rads;
        break;
        case 19:
                lam_zero = -69*rads;
        break;
        case 20:
                lam_zero = -63*rads;
        break;
        default:
                printf("returning control to _main with null value\n");
                *zone = 0;
                return(0);
        break;
};

cosphi = cos(phi);
ep_square = e_square/(1-e_square);
n = alpha/sqrt(1 - (e_square * pow(sin(phi),2.0)));
t = pow(tan(phi),2.0);
c = ep_square * pow(cosphi,2.0);
a = cosphi * (lambda - lam_zero);

/*      ----->phi_zero always = 0 (as far as I can tell), so mo will always
 *            equal zero.  Why calculate it? Just in case, here is the
 *            formula from Snyder.
 *
 *    mo = 111132.0894 * phi_zero - 16216.94 * sin((2 * phi_zero)*rads) +
 *    17.21 * sin((4 * phi_zero)*rads) - 0.02 * sin((6 * phi_zero)*rads);
 */
mo = 0.0;
phi /= rads;
m = 111132.0894 * phi - 16216.94 * sin((2 * phi)*rads) +
17.21 * sin((4 * phi)*rads) - 0.02 * sin((6 * phi)*rads);
newx = (k_zero * n) * (a + (((1 - t + c) * pow(a,3.0))/6) + (((5 - (18 * t)
+ pow(t,2.0) + (72 * c) - (58 * ep_square)) * pow(a,5.0))/120));
*x = (int) ((newx + 500000) * 100);
newy = k_zero * (m - mo + n*tan(phi*rads) * (pow(a,2.0)/2 + (5 - t + 9*c
+ 4*pow(c,2.0)) * pow(a,4.0)/24 + (61 - 58*t + pow(t,2.0) + 600*c -
330*ep_square) * pow(a,6.0)/720));
*y = (int) (newy * 100);

return(1);
}


#ifdef TEST        /*   -----> this is a test harness for ll_utm.c  */
main()
{
int     x,y,zone;

printf("Welcome to test harness of ll_utm.c\n");
printf("\nenter latitude :");
scanf("%d",&y);
printf("enter longitude:");
scanf("%d",&x);
```

```
if (!ll_utm(&y,&x,&zone))
        printf("\n** reminder!!! - no changes have been made to coordinates\n");
printf("\nx coordinate = %d\n", x);
printf("y coordinate = %d\n", y);
printf("zone = %d\n", zone);
}
#endif
```

**utm_ll.c**

```
/***************************************************************************
*******/
/*      ----->function utm_ll: accepts parameters lon (y) and lat (x), which
 *            are expected to be double precision UTM coordinates passed by
 *            value (!they will be changed!).  The parameter zone is also
 *            required, but is only passed by reference.
 *            **NOTE - this procedure has been written to handle ares in
 *            the coterminus U.S., and will fail outside of the northern
 *            hemisphere, or outside of zones designated below.
 */


#include <math.h>
#include <stdio.h>
#include "joe.h"
#define alpha 6378206.40      /*      ----->set parameters for Clarke    */
#define e_square 0.00676866 /*          1866 ellipsoid            */
#define k_zero 0.9996


int utm_ll(lon,lat,zone)
double *lon,*lat;
int     zone;
{
double ep_square,phi_zero,lam_zero,mo,m,mu,rads,
        e_one,phi,c_one,t_one,n_one,r_one,d_one;


/*      ----->set origin to the equator.
 */
        phi_zero = 0;

/*      ----->set rads as conversion factor from degrees to radians.
 */
        rads = 180/PI;

/*      ----->subtract 500,000 from the x value supplied to the routine (false
            easting correction).
 */
        *lon -= 500000;

/*
set semi_minor axis for conversion according to zone parameter
 */

switch (zone)  {
        case 10:
                lam_zero = -123/rads;
        break;
        case 11:
                lam_zero = -117/rads;
        break;
        case 12:
                lam_zero = -111/rads;
        break;
        case 13:
                lam_zero = -105/rads;
        break;
        case 14:
                lam_zero = -99/rads;
        break;
        case 15:
                lam_zero = -93/rads;
        break;
        case 16:
                lam_zero = -87/rads;
        break;
        case 17:
                lam_zero = -81/rads;
        break;
        case 18:
                lam_zero = -75/rads;
        break;
        case 19:
                lam_zero = -69/rads;
        break;
        case 20:
                lam_zero = -63/rads;
        break;
        case 21:
                lam_zero = -57/rads;
        break;
        case 22:
                lam_zero = -51/rads;
        break;
        case 23:
                lam_zero = -45/rads;
        break;
        default:
                return(0);
};


/*
The following calculations implement an algorithm for
calculating the inverse Transverse Mercator (Ellipsoid) projection. Taken from
pp. 68-69 of U.S. Geological Survey Bulletin 1532 (1983) - "Map Projections
Used by the U.S. Geological Survey", Author - John Snyder.
 */


mo = (111132.0894 * phi_zero) - (16216.94 * sin((2 * phi_zero)/rads)) +
(17.21 * sin((4 * phi_zero)/rads)) - (0.02 * sin((6 * phi_zero)/rads));

ep_square = e_square/(1 - e_square);
m = mo + (*lat/k_zero);
```

```
e_one = (1 - sqrt(1 - e_square))/(1 + sqrt(1 - e_square));
mu = m/(alpha * (1 - (e_square/4) - (3 * (pow(e_square,2.0)/64))
 - (5 * (pow(e_square,3.0)/256))));
phi = mu + ((3 * (e_one/2) - (27 * (pow(e_one,3.0)/32))) * sin(2 * mu))
+ (((21 * (pow(e_one,2.0)/16)) - (55 * (pow(e_one,4.0)/32))) * sin(4 * mu))
+ (((151 * (pow(e_one,3.0)/96))) * sin (6 * mu));
c_one = ep_square * pow(cos(phi),2.0);
t_one = pow(tan(phi),2.0);
n_one = alpha/sqrt(1 - e_square * pow(sin(phi),2.0));
r_one = alpha * (1 - e_square)/pow((1 - e_square * pow(sin(phi),2.0)),1.5);
d_one = *lon/(n_one * k_zero);
*lat = phi - (n_one * (tan(phi)/r_one)) * ((pow(d_one,2.0)/2) - (5 + (3*t_one)
+ (10*c_one) - (4*pow(c_one,2.0)) - (9*ep_square)) * (pow(d_one,4.0)/24))
+ ((61 + (90*t_one) + (298*c_one) + (45*pow(t_one,2.0)) - (252 * ep_square)
- (3 * pow(c_one,2.0))) * (pow(d_one,6.0)/720));
*lat *= rads;

*lon = lam_zero + (d_one - ((1 + (2 * t_one) + c_one) * (pow(d_one,3.0)/6))
+ ((5 - (2 * c_one) + (28 * t_one) - (3 * pow(c_one,2.0)) + (8 * ep_square)
+ (24 * pow(t_one,2.0))) * (pow(d_one,5.0)/120)))/cos(phi);
*lon *= rads;
*lon = ABS(*lon);
return(1);
}


#ifdef TEST          /*   -----> this is a test harness for utm_ll.c  */
main()
{
double  x,y;
int        zone,intx,inty,check;

printf("Welcome to test harness of utm_ll.c\n");
printf("\n\7enter x coordinate:");
scanf("%lf",&x);
printf("\7enter y coordinate:");
scanf("%lf",&y);
printf("\7enter UTM zone:");
scanf("%d",&zone);
check = utm_ll(&x,&y,zone);
if(check)
           {
           intx = x * 1000000;
           inty = y * 1000000;
           printf("\nlatitude (dd) = %d\n", inty);
           printf("longitude (dd) = %d\n", intx);
           }
else
           printf("conversion failed!!\n");
}
#endif
```

**alber_fwd.c**

```
/*       ----->function albers_ffwd:  accepts parameters x (latitude) and y
 *           (longitude), which must be double precision variables
 *           in decimal degree format.  It transforms x and y to meters by
 *           the algorithm given by Snyder (USGS Bulletin 1532 - p.96-97) for
 *           the Albers equal-area conic projection (Ellipsoid form).
 *           **NOTE: this function has been written for areas in N. America,
 *         with standard parallels of 23.30 N and 45.30 N.  The origin
 *           is 23.00 N and 96.00 W.  Any other parameters will require
 *           modifying the source code (refer to albers_fwd.c).
 */


#include <math.h>
#include <stdio.h>
#define ABS(x)                  (((x) < 0) ? -(x) : (x))
#define ANTI_ABS(x)             (((x) < 0) ?  (x) : -(x))
#define alpha    6378206.40     /*  ----->set parameters for Clarke    */
#define e_square 0.00676866     /*       1866 ellipsoid               */
#define e        .0822719
#define rads     PI/180         /*  ----->factor for radian conversions */
#define lam0     -96.00
#define n        .6029035
#define c        1.3491593857
#define r0       9929079.569251543



int albers_ffwd(x,y)
double  *x,*y;

{
double  q,r,theta,phi,lam,sinphi;

phi = ABS(*y);
lam = ANTI_ABS(*x);
sinphi = sin(phi*rads);
q = (1 - e_square)*(sinphi/(1 - e_square*pow(sinphi,2.0)) - (1/(2*e)) *
log((1 - e*sinphi)/(1 + e*sinphi)));
r = alpha*sqrt(c - n*q)/n;
theta = (n*(lam - lam0))*rads;
*x = r * sin(theta);
*y = r0 - r*cos(theta);
return(1);
}


#ifdef TEST         /*  -----> this is a test harness for albers_ffwd.c  */
main()
{
double  x,y;

printf("Welcome to test harness of albers_ffwd.c\n");
printf("\nenter latitude :");
scanf("%lf",&y);
printf("enter longitude:");
scanf("%lf",&x);
printf("\nx coordinate = %f\n", x);
printf("y coordinate = %f\n", y);
if (!albers_ffwd(&x,&y))
        printf("\n** reminder!!! - no changes have been made to coordinates\n");
printf("\nx coordinate = %f\n", x);
printf("y coordinate = %f\n", y);
}
#endif
```

# CONVERTING THE LEE COUNTY FLORIDA DATASET INTO ARC/INFO FORMAT:

(NOTE: This document assumes you have a version of Arc/Info which supports the TIGERARC function.)

## PREPROCESSING

Delete the three-line header (up to and including the third carnage return) from each file. This can be done with a text editor or inside of a program.

The data are now in TIGER **prototype** format. TIGER **prototype** is the file format used by the Census Bureau prior to the current TIGER **precensus** format.

Older versions of Arc/Info will only support the prototype format; newer versions are able to convert both prototype and precensus formats.

## CONVERSION

Two files are needed to produce an Arc/Info coverage: (these files have the same names followed by a ".1" or ".2" extension) Type 1 files contain information on the nodes and attributes of each arc in the coverage. Type 2 files (shape files) contain the coordinates of all the vertices belonging to each arc. An arc need not have any vertices-two nodes define an arc.

On the older versions of Arc/Info, type:

```
TIGERARC [outcover] [file.1] [file.2]
```

On newer versions (those that handle both TIGER formats), type:

```
TIGERARC PROTO [outcover] [file.1] [file.2]
```

## CORRECTION

Before deleting the original data files, verify that the conversion was successful by displaying the coverage in Arcplot or Arcedit.

Occasionally a few bad arcs are generated with extreme (and impossible) coordinate values. When this happens the plot of the coverage may look like a single point or a small cluster of points along with one or two longer lines (usually straight). If this is the case, zoom in on the small cluster (**MAPEXTENT *** works nicely) and replot the coverage. If the zoomed coverage looks correct (i.e. if it doesn't look like random garbage) go in to Arcedit and delete the large, spurious arcs. Then delete the tics and add new tics closer to the actual coverage (the cluster of points). A replot of the coverage should now produce a map which looks legitimate.

If the above strategy fails to produce a good coverage, check the original data files and make sure that the headers were completely deleted (and that nothing beyond the headers was deleted).

## CREATING TOPOLOGY

The coverage which has just been created lacks topology. You must **CLEAN** and **BUILD** the coverage in order to create arc and/or polygon topology. Refer to the appendices of the Arc/Info Volume 2 manual (command references) for a more detailed discussion on processing TIGER coverages.

## THE NCGIA/US CENSUS MULTIPLE REPRESENTATIONS DATA SET: LEE COUNTY, FLORIDA

SOURCE: NCGIA, Department of Geography
University of California
Santa Barbara CA 93106-4060
Phone, 805/893-8224
FAX: -8617

Price: $15 formatting and mailing. Purchaser provides 7 DOS 3.5" diskettes.

FORMAT: TIGER prototype. This is an ASCII format used by the Census bureau prior to the current TIGER pre-census format. The entire data set resides in compressed form on 7 high density 3.5" diskettes. The total uncompressed size of the data set is approximately 43 Megabytes.

DESCRIPTION: This is a multi-scale, multi-agency database, which was developed for use in teaching and research, primarily in the subject area of multiple representations. The dataset consists of vector data from various sources covering all of Lee County, Florida, and surrounding areas. The components of the database are as follows:

1. Tiger/Line Data: Various data products from the US Census Bureau, including the entire TIGER file for Lee County (12 Meg.)

2. USGS Digital Line Graph (DLG) Data

3. USGS Land Use/Land Cover Data

4. National Ocean Service (NOS) Shoreline Data

Documentation for this data set is included in the NCGIA Technical Report #90-4: The NCGIA/US Census Multiple Representations Data Set Project-- Technical Report on Pilot Project: Lee County, Florida, May, 1990.

Current as of July 1991.