EPIC-VIEW: A FULLY INTEGRATED SPATIAL TOOL FOR MODELING SOIL EROSION AND AGRICULTURAL CROP PRODUCTIVITY

By

ANOOP GOVIL

Bachelor of Engineering

Bangalore University

Karnataka, India

1992

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
July, 1996

EPIC-VIEW: A FULLY INTEGRATED SPATIAL TOOL FOR MODELING SOIL EROSION AND AGRICULTURAL CROP PRODUCTIVITY

Thesis Approved:

Thesis Adviser

Dean of the Graduate College

ACKNOWLEDGMENTS

I wish to express my sincere appreciation to my major adviser, Dr. Mitchell L. Neilsen for his guidance, supervision, encouragement and help for the completion of my thesis work. His patience and constructive ideas helped me make this thesis work an enjoyable and memorable experience. I consider it a privilege to have worked under his supervision.

I would like to express my sincere thanks to Dr. Blayne E. Mayfield and Dr. K.M. George for serving on my graduate committee. Their support and invaluable suggestions, have helped me to improve the quality of this work. I would like to thank Dr. David A. Waits, Department of Geography, for his constructive ideas and support which proved to be vital during the development stages. I would like to thank Dr. Mitchell L. Neilsen and the Computer Science Department for providing me with this research opportunity and their generous financial support.

My greatest appreciation, thanks and love to my parents Mr. M.L. Govil and Mrs. Sudha Govil and Amit for all the love, support and inspiration that they have given me. Also, thanks to all my friends for their support and much needed help.

TABLE OF CONTENTS

LIST OF FIGURES

Figure Page

CHAPTER I

INTRODUCTION

Geographic information systems (GIS) have the potential to aid agricultural producers in determining cause and effect relationships between management and production, to project production, and to account for spatial and temporal differences within specific agricultural fields. Currently, only a few producers are utilizing the true analytical power of GIS and computer simulation models, partly because the software developed to date loosely links the GIS with the simulation software. This makes usage of the software more labor intensive. A need exists for a fully integrated, user-friendly GIS-modeling system that allows producers to efficiently simulate soil erosion, plant growth and related process, and economic components for assessing the cost of erosion and comparative results of using different management techniques.

GIS is emerging as an important tool in modeling. An important feature of environmental modeling is that all basic units (water, soil, and chemicals) have spatial distributions, and thus can be linked with the GIS. GIS software has been developed to capture, manipulate, process, and display spatial or georeferenced data. GIS linkage to a model varies from a loose coupling to a complex integration (which is highly desirable). GIS is frequently used to prepare spatially distributed input data which is then passed to the linked model which processes it, and to display and probably analyze model results. Integration of a GIS with a model minimizes the problems of data management, as most

of the data can directly be extracted from the GIS itself, and offers the capability to integrate spatial and modeling process into a single interactive system.

In this thesis we develop a single interactive system, called EPIC-View, that fully integrates the impact calculator EPIC with ArcView, Version 2.1 (Environmental Systems Research Institute, 1994b). There are numerous benefits of using this type of fully integrated tool. The system is easy to use and more efficient (as all the operations are automated); spatial data is extracted directly from the existing GIS; and spatial output is displayed using the same tools by which the input data is displayed spatially. EPIC is tightly integrated with ArcView; the user is able to execute EPIC and display spatial output data using ArcView.

1.1 Problem Statement

The objective is to develop a single interactive system, called EPIC-View, that fully integrates the Erosion/Productivity Impact Calculator (EPIC) (Sharpley and Williams, 1990) with ArcView®, Version 2.1 (Environment Systems Research Institute®, 1994b). The steps to be completed are outlined below:

1. The EPIC model requires both spatial and farm management-related data as input. Data required for modeling, originated from following sources :

Within the Fort Cobb Reservoir watershed, two quarter-sections located in Caddo County, Oklahoma, was selected as the study area for this project. A GIS-oriented database was developed to incorporate data needed for modeling. Site-specific data

required by the model was procured from an already identified producer's field records. These include soil fertility, crop rotation practices, conservation program, farm chemicals application, and tillage systems. Additional model input parameters were compiled from public domain data sets. Some of these datasets include soils, elevation, slope, water bodies (streams, water holes, etc.). Derived coverages from the primary GIS coverages were also developed.

The resultant data, which is used for modeling, consists of gridded coverages, which are overlaid homogeneous units consisting of GIS attributes, such as soil, crop, elevation, slope, etc. Most of the data required for the EPIC input form can be derived directly from the GIS. A graphical user interface, based on Visual Basic®, Version 4.0 (Microsoft® Corp.) is developed for entering other data that cannot be derived from the GIS.

2. The EPIC model is linked to ArcView using ArcView's scripting language called Avenue® (Environmental Systems Research Institute, 1994a). Avenue scripts provide a customized interface, other scripts are associated with various controls such as buttons, menu options, etc. In this way, the EPIC model is invoked directly from ArcView. A user can delineate management zones on the field coverage and enter management practices through the user interface. The GIS attributes are extracted from the existing GIS database. Finally EPIC is invoked after specifying the output options (described in

Section 4.1.3) for EPIC output files. All steps are automated by invoking associated scripts.

3. Spatial output results from the EPIC model are also linked with the GIS, by parsing the output, generated by EPIC, into ArcView readable formats. This output is loaded back into the GIS. In this way, the user is able to visualize some of the tabular output data as spatial graph using ArcView and consequently make better recommendations based on that data.

4. Finally, the new interactive modeling tool is tested by running the model on the homogeneous units, created as a result of overlaying various GIS attributes.

CHAPTER II

LITERATURE REVIEW

2.1 Erosion/Productivity Impact Calculator (EPIC)

2.1.1 About the Model

In the early 1980's, teams of USDA Agriculture Research Service (ARS), Soil Conservation Service(SCS), and Economic Research Service (ERS) scientists developed EPIC to quantify the costs of soil erosion, and the benefits of soil erosion research and control in the United States. Led by Dr. J.R. Williams, ARS scientists were responsible for model development. SCS and ERS staff collaborated on the model development and took leading roles in soil and weather dataset development, validation, and interface creation for economic models.

In the late 1980's, Texas Agricultural Experiment Station scientists became involved in the model support, documentation, database development and technology transfer.

EPIC is designed to be:

- capable of simulating the relevant biophysical processes simultaneously, as well as realistically, using readily available inputs and, where possible, accepted methodologies;

- capable of simulating cropping systems for hundreds of years because erosion can be a relatively slow process;
- applicable to a wide range of soils, climates and crops;
- efficient, convenient to use, and capable of simulating the particular effects of management on soil erosion and productivity in specific environments.

The model uses a daily time step to simulate weather, hydrology, soil temperature, erosion-sedimentation, nutrient cycling, tillage, crop management and growth, pesticide and nutrient movement with water and sediment, and field-scale costs and returns.

2.1.2 Model Components

In EPIC the major biophysical processes simulated are called components. EPIC consists of following ten major components:

Weather: Daily rain, snow, maximum and minimum temperatures, solar radiation, wind and relative humidity can be based on measured data and/or generated stochastically.

Hydrology: Runoff, percolation, lateral subsurface flow, and snow melt are simulated. Any one of four methods can be used to estimate potential evapotranspiration.

Erosion: EPIC simulates soil erosion caused by wind and water. Sheet and rill erosion/sedimentation result from runoff from rainfall, snow melt, and irrigation.

Nutrient Cycling: The model simulates nitrogen and phosphorus fertilization, transformations, crop uptake and nutrient movement. Nutrients can be applied as mineral fertilizers, in irrigation water, or as animal manures.

Pesticide Fate: The model simulates pesticide movement with water and sediment as well as degradation on foliage and in the soil.

Soil Temperature: Soil temperature responds to weather, soil water content, and bulk density. It is computed daily in each soil layer.

Tillage: Tillage equipment affects soil hydrology and nutrient cycling. The user may change the characteristics of simulated tillage equipment, if needed.

Crop Growth: A single crop model capable of simulating major agronomic crops, pastures, and trees is used. Crop-specific parameters are available for most crops. The user may adjust or create new sets of parameters as needed. The model can also simulate crops grown in complex rotations and, in certain cases, in mixtures.

Crop and Soil Management: The EPIC model is capable of simulating a variety of cropping variables, management practices and naturally occurring processes. These include different crop characteristics, plant populations, dates of planting harvest, fertilization, irrigation, artificial drainage systems, tillage, runoff control with furrow dikes and other methods, liming, and pest control. The model can also gauge the effects of such varied management practices, as whether the crop is harvested for grain or fodder or if it is grazed or burned.

Economics: A simple accounting package is included to calculate the cost of inputs and the value of returns.

2.1.3 Selected EPIC Applications

Agricultural systems typically evolve over long periods of time in response to climate, soils, agricultural technology, socio-economic conditions and other factors. Long-term sustainability of such systems requires that they:

- be economically sound in the local socio-economic context,
- conserve and/or protect crucial soil and water resources, and
- be capable of adapting to the changing social, economic, and natural environments.

EPIC is designed to help decision makers analyze alternative cropping systems and project their socioeconomic and environmental sustainability. This section highlights several studies in which the model has been used to evaluate crop productivity, risk of crop failure, degradation of the soil resource, impacts on water quality, response to different input levels and management practices, response to spatial variation in climate and soils, and long-term changes in climate.

Accurate simulations of crops yields are necessary for most applications of models like EPIC. Studies like those which follow typically contain preliminary activities to test model sensitivity. In addition, model developers continually monitor the effects of model improvements on simulation of yields and other important outputs.

Crop Productivity: Dr. J.R. Williams (1989) evaluated EPIC's ability to simulate yields of maize, wheat, rice, sunflower, barley and soybeans using a total of 227 measured yields reported by independent research groups around the world. For these crops, mean simulated yields were always within 7% of mean measured yields. For 118 comparisons of measured and simulated maize yields, mean measured yield and its standard deviation were 103 bushels per acre and 49 bushels per acre, respectively. The measured and simulated means were not significantly different at the 95% confidence level. He also demonstrated that EPIC can accurately simulate maize responsed to irrigation at locations in the western USA and to fertilizer nitrogen in Hawaii.

Soil Degradation: EPIC was originally designed to estimate the loss of crop production due to soil erosion. For the RCA analysis, EPIC simulation runs of 100 years were made for each of over 13,000 combinations of crops, soils, climates, tillages, and conservation practices. Simulation results were used by a large linear programming model to assess the impacts of soil conservation practices and erosion on agricultural production of the USA. EPIC was also used to demonstrate that, even though the effects of soil erosion on crop productivity may be small for long period, high rates of erosion can drastically shorten the productive life of soils.

Input Levels and Management Practices: Cabelguenne, et al. (1988, 1990) used EPIC in southern France to simulate growth and yield of corn, grain sorghum, sunflower, soybean, and wheat grown in rotations over a five-year period. Each crop had three levels of fertilizer, irrigation, and tillage. The root mean square error of simulated grain yields ranged from 15 bushels per acre for sunflower to 26 bushels per acre for corn. Mean

simulated yields were not significantly different than mean measured yields for summer crops, and for individual plots, simulated and measured yields were within 20% of each other for 81% of comparisons.

Dyke, et al.(1990) compared simulated and measured yields for a total of 204 treatment years for the Southern Coastal Plain and Southern High Plains of Texas. Crops included maize, grain sorghum, and cotton. Tillage systems, irrigation, and crop rotations also varied. Simulated yields were within 20% of mean measured yields for 70 and 90% of treatment-years for the Coastal Plain and High Plains, respectively. Simulated yields were within the 95% confidence interval of measured yields for 69 and 88% of the treatment-years for the two sites. AUSCANE, a version of EPIC adapted to Australian sugarcane (Jones, et al., 1989), was used to demonstrate the importance of irrigation in reducing the risks in sugarcane production near Mackay, Queensland.

Segarra (1989) used EPIC to evaluate optimum nitrogen fertilizer rates for cotton in the Southern High Plains of Texas. Nitrogen and cotton prices were found to affect optimum fertilizer rates, so the use of decision rules based on these prices could improve the cash flow of producers.

Because it can simulate a variety of important agricultural practices, the model has also been used successfully to estimate crop fertilizer requirements, nutrient transport in runoff, soil and fertilizer phosphorus dynamics, the effect of furrow diking on crop yields, and low-input legume-based crop rotations.

Recent addition of pesticide components enable the model to simulate movement of pesticides and nutrients toward ground and surface waters, both in solution, and, as appropriate, attached to sediments. This capability provides agricultural managers and policy makers with a powerful, comprehensive tool to assess simultaneously the impacts of management and soil on crop production, risks, and soil and water resources.

Response to Climates and Soils: Arnold and Jones (1987) evaluated EPIC's sensitivity to soil, climate and rotation effects on crop (maize, soybean, wheat, barley, peanut, and hay) productivity and fertilizer nitrogen requirements. They concluded that EPIC can be used to evaluate previously untested combinations of soil, climate, and crop management, thereby reducing the amount of site-specific research needed to assess improved agricultural technology.

Jones, et al.(1989) demonstrated that the AUSCANE version of EPIC can accurately simulate the effect of different climates and management practices on sugarcane yields and sugar concentrations throughout Australia's sugar-growing areas.

EPIC has been widely used by agricultural economists and others to simulate the effects of weather, climate, and crop management practices on the crop productivity, risk, and degradation of the soil resource. For example, Lee and Lacewell (1990) used it to optimize selection of irrigated crops and associated withdrawal of groundwater of the Texas Southern High Plains, with and without farmer participation in government farm programs. They concluded that strategies that reduce risk would also reduce irrigated area and groundwater extraction. In contrast, participation in farm programs would increase extraction rate.

Lee and Lacewell (1989) used EPIC to simulate yields, wind erosion, and net returns in the Texas Southern High Plains for several cropping systems and irrigation options, with and without participation in government farm programs. Results from EPIC were analyzed with a farm-level economic optimization model. The study indicated that compliance with the base acreage provisions of the farm program limits adoption of profitable, soil conserving cropping systems.

They also used EPIC to evaluate crop yield, erosion and net returns for twelve alternative dryland crop rotations in the Southern High Plains of Texas, with and without participation in federal farm programs. They concluded that, with participation, cotton is an essential part of profitable dryland farming systems. However, cotton is associated with high rates of soils erosion and, thus, requires rotation with wheat to reduce the amount of erosion and comply with the farm program. Continuous cotton planted after a winter wheat cover crop terminated with herbicide late in winter appears to be a viable cropping system.

Vicien (1989) used EPIC to construct production functions for wheat grown in Argentina and France. Such functions could then be used to optimize management practices considering the interacting effects of soils, climates, possible production practices, input costs and commodity prices.

Climate Change: In addition to regional and farm-scale economic analysis, EPIC has been used to assess the effects of short and long-term climatic changes. The U.S. Department of Agriculture used it during the summer of 1988 to predict the effects of that year's severe drought on U.S. crop production.

EPIC and SOYGRO(a soybean growth model) were used to predict the effects of war-induced "nuclear winter" on crop growth and yields in the United States (Jones et al. 1988). Four timing scenarios and three severity scenarios were simulated. Similar results were obtained with the two models for effects on soybean yields, suggesting that EPIC behaves comparably to a more complex physiological model of soybean growth and development under extreme conditions of temperature and solar radiation.

Robertson et al. (1987, 1990) used EPIC to predict the impacts of CO2 and climate change scenarios on crop yields, soil erosion, and farm management for the U.S. Great Plains, Corm Belt, and Southeast. Recent model improvements permit more accurate simulation of the effects of CO2 and climate change on hydrology and crop growth.

Resources for the Future used EPIC to simulate the effects of changing CO2 and climate on crop yields and farm profitability in Missouri, Iowa, Nebraska, and Kansas. In that study, the warmer and drier weather of the 1930s was used as a surrogate for future climate (Easterling III, et. al., 1991).

Water Quality: Some of the most recent improvements in EPIC have enhanced its ability to simulate the impacts of cropping systems on water quality. Components of the GLEAMS (Groundwater Leaching Effects on Agricultural Management Systems) model have been added to permit EPIC to simulate degradation and movement of pesticides in the soil. The fertilization and nutrient cycling components have also been improved to enable simulation of a variety of animal manures, fertigation, and contamination of irrigation water with mineral nitrogen. The model is now being used in the United States

and Europe to assess the impacts of "best management practices" on parameters of surface and ground water quality.

These EPIC applications could have far reaching effects on global agricultural practices. The flexibility of the model permits farm managers, policy makers and scientists from all over the world to tailor their cropping systems to particular combinations of natural resources, socioeconomic conditions, and management possibilities. It allows environmental quality to be considered, as well as productivity, cost and profitability.

2.1.4 EPIC's Universal Text Integration Language (UTIL)

UTIL is an on-line, input dataset editor which comes along with EPIC. UTIL is a companion interface program that helps users build EPIC data sets, execute the model, and display the results. UTIL has its standalone environment. It facilitates data entry for creation of EPIC input dataset. It provides on-line description of each data variable, its legal ranges, although a user is allowed to enter value outside the legal ranges. EPIC requires that each data variable in the input dataset have a particular position in the input dataset which is a text file with an extension ".dat". UTIL facilitates this by automatically placing each variable value in its respective place. UTIL has both interactive mode and batch mode of creating dataset. UTIL supports the use of a batch file with an extension ".utl" which can contain UTIL commands recognized by UTIL and also variable abbreviations recognized by UTIL along with their values spaces by atleast one space. This is a powerful feature of UTIL as it places the value of each variable in its respective

position in the dataset though it is read from the "batch" file. A user does not have to worry about the placement of variables' values as UTIL takes care of that, which eliminates any possibility of input dataset with a wrong format. UTIL also has features of displaying the output files, generated by EPIC, such as ".epy", ".epm", ".epy" files depending upon what output option a user selected. These files give summary of various variables' values every day/month/year (as selected by the user). UTIL serves as an editor to display these files and also provide on-line help for each of the variables found in these files. All the driver files accompanied by EPIC can be edited by UTIL. Hence UTIL provides a total data entry/maintenance environment for EPIC.

2.2 ArcView®, Version 2.1 (Environmental Systems Research Institute®, 1994b)

2.2.1 About ArcView

ArcView is a powerful, easy-to-use tool that brings geographic information to the desktop. ArcView gives users the power to visualize, explore, query and analyze data spatially. ArcView comes with a useful set of ready-to-use sample data. If data in the ARC/INFO® format is available, a user will be able to use ArcView to access all of this data, including vector coverages, map libraries, grids, images and event data.

Working spatially:

ArcView can be used to work spatially. Tabular data, such as dBASE files and data from database servers, can be loaded into ArcView so that a user can display, query, summarize, and organize this data geographically.

Views:

With ArcView a user works with geographic data in interactive maps called views. Every view features a geographic 'Table of Contents', making it easy to understand and control what's displayed.

Tables:

If a user clicks on features on a view, their records highlight in the table showing him their attributes, or selects records in the table, the features they represent highlight on the view. Tables also have a full range of features for obtaining summary statistics, sorting and querying.

Charts:

Charts offer a powerful business graphics and data visualization capability that is fully integrated into the geographic environment. A user can click on features on a view to add them to the chart. ArcView allows a user to work simultaneously with geographic, tabular and chart representations of his data.

Layouts:

Layouts allow a user to create high quality, full color maps by first arranging the various graphic elements on-screen the way he wants them. Layouts have a live link to the data they represent. When a user prints a layout, any changes to the data are automatically included, so everything on his map will be up-to-date.

Scripts:

ArcView scripts are macros written in Avenue, ArcView's programming language and development environment. With Avenue a user can customize almost every aspect of ArcView, from adding a new button to run a script a user writes, to creating an entire custom application that he can distribute.

Projects:

All the components of a user's ArcView session: views, tables, charts, layouts, and scripts are stored in one file called a project. ArcView's Project window shows a user, the contents of his project and makes it easy to manage all his work.

2.2.2 ArcView Components

1. View:

A view is an interactive map that lets user display, explore, query and analyze geographic data in ArcView.

A view defines the geographic data that will be used and how it will be displayed, but it doesn't contain the geographic data files themselves. Instead, a view references these source data files. Thus, a view is dynamic, because it reflects the current status of the source data. If the source data changes, a view that uses this data will reflect the change the next time the view redraws.

The same data can be displayed on more than one view. Different users may have different views on the same data. A different view of the data can be created for each application a user has.

A user can use existing views or create new ones. Views can be modified as the needs change. Views can also be created for others to use, in which case some or all of the views' contents might be locked so they can't be modified. With Avenue a user can create custom functions, user interfaces and applications based on views.

A view is actually a collection of themes. A theme represents a distinct set of geographic features in a particular geographic data source. For example, a view of a country might have one theme representing cities, one theme representing roads, one representing rivers, etc.

A view is displayed inside a window. A user can resize a view's window and zoom in or out on a view to display a particular area or extent.

View's window contains 'Table of Contents'. A view's Table of Contents lists the themes in the view and lets user control how the themes are displayed and the order in which they are drawn. A user can turn individual themes on or off, if needed.

2. Table:

A table lets a user work with data from various tabular data sources in ArcView. A user can display, query and analyze data in tables. Records can be highlighted in tables by selecting geographic features displayed on views, and vice versa. Tables can be displayed on a view to reveal the geography of the data. Charts can be created from tables to visualize trends, patterns and distributions.

An ArcView table references the tabular data source it represents, but doesn't contain the tabular data itself, hence tables are dynamic, because they reflect the current

status of the source data they are based on. If the source data changes, a table based on this data will reflect the change the next time a user opens the project containing this table. A user can also refresh the table at any time to see the current state of their source data.

Some tables can also be edited, depending on the data source for user's table. All edits are written back to the source data file.

Spatial data sources such as ARC/INFO coverages have attribute tables containing descriptive information about the geographic features they contain. A user can use a view containing themes that represent these spatial data sources and have access to their attribute tables. ArcView manages the relationship between themes and their attribute tables, these tables do not need to be loaded into ArcView separately.

A user can add dBASE, INFO, and tab or comma delimited text files into ArcView as tables.

From within ArcView a user can connect to a database server, such as Oracle or Sybase, and run an SQL query to retrieve records from it as a table. ArcView stores the definition of the SQL query, the user used, rather than the records themselves.

3. Chart:

A chart is a graphic presentation of tabular data that provides an additional visual representation of the attributes associated with geographic features. A user can use a chart to display, compare, and query geographic and tabular data effectively.

A chart references tabular data in an existing ArcView table in the user's project, and defines how it will be displayed. Charts are also dynamic because they reflect the current status of the data in the table. If there is a change in the source data on which the table is based, this change will automatically be reflected in both the table and the chart the next time a user opens the project that contains them. If the table is edited, the chart will reflect the edit.

A chart can represent all or a selected subset of records in a table. Records can be selected from the table, and also, if the table is an attribute table belonging to a theme, by selecting the theme's features on a view. If the selected set of records changes, the chart will reflect the new selection.

The same tabular data can be displayed on more than one chart.

4. Layout:

A layout is a map that lets a user display views, charts, tables, imported graphics, and graphic primitives. The layout is used to prepare these graphics for output from ArcView.

A layout defines what data will be used for output and how they will be displayed. A layout is dynamic because it allows user to make specific graphics which reflects the current status of the data. If the data in a view changes, the layout reflects the change.

Different layouts can be created based on same data. Each layout can be considered a different way of presenting the data. Using Avenue a user can create custom functions, user interfaces and cartographic templates that will assist in creating output.

Layout provides standard graphics and operations. These graphics are drawn using the Draw tool and include points, lines, polygons, polylines, rectangles, and circles. Layout also contains objects specific to the ArcView environment, including frames containing ArcView views, charts, and tables, and ancillary objects such as legends and scale bars.

5. Scripts:

A script is the component of an ArcView project that contains Avenue code. ArcView scripts group together the means to accomplish three general objectives: automate tasks, add new capabilities to ArcView, and build complete applications.

All of ArcView can be considered a collection of scripts. Every control that a user uses in ArcView, has an associated internal or system script. A user can see the names of the scripts associated with a control in the Customize dialog box and can examine the contents of a system script by loading a system script into a new script.

ArcView has a Script Editor where a user can create a script. If a user uses other text editor, once he has written the code, he must load it into a project's script. A user can compile, debug, and run the script from within the Script Manager.

A user can use ArcView's customization environment to associate a compiled script with a control or with an event, such as starting up or shutting down a project.

2.2.3 ArcView Scripting Language Avenue®

Avenue is the programming language and development environment that's part of ArcView. Avenue is fully integrated with ArcView and the work, a user does, will run on any of the platforms for which ArcView is available. There are many uses for Avenue: a user can use Avenue to customize working with ArcView; or direct ArcView to perform a specific task that needs to be done; or a complete application can be developed, that works along with ArcView's graphical user interface.

ArcView provides the necessary customization and language environment so a user can work with Avenue. A user can create the graphical user interface as per the requirements, establish some initial properties for the graphical controls that a user will interact with, fine tune the behavior and appearance of those controls, and write Avenue code that responds to what goes on in the interface, created. In addition, scripts written in Avenue, can be linked to events such as starting up and shutting down a project.

2.3 Visual Basic®, Version 4.0 (Microsoft® Corp.)

Microsoft Visual Basic allows a user to create quick applications for Microsoft Windows operating systems. The Visual Basic programming system allows a user to create useful applications that fully make use of the graphical user interface (GUI).

Visual Basic provides a user, appropriate tools for the different aspects of GUI development. A user can create a graphical user interface for different applications by drawing objects in a graphical way and set properties on these objects to refine their

appearance and behavior. Generated interface can react to a user by the attached code that responds to events that occur in the interface.

A user can create, full-featured applications. Different features of Visual Basic, are as follows:

- Data access features allow a user to create databases and front-end applications for popular database formats.
- OLE allows a user to use the functionality provided by other applications, such as Microsoft Word® for Windows word processor, Microsoft Excel® spreadsheet, and Microsoft Project® business project planning system.
- User's finished application is a ".EXE" file that uses a run-time dynamic-link library (DLL).

CHAPTER III

DESIGN AND IMPLEMENTATION

3.1 Main Menu for EPIC-View

The main menu of EPIC-View has options as shown in figure 1.

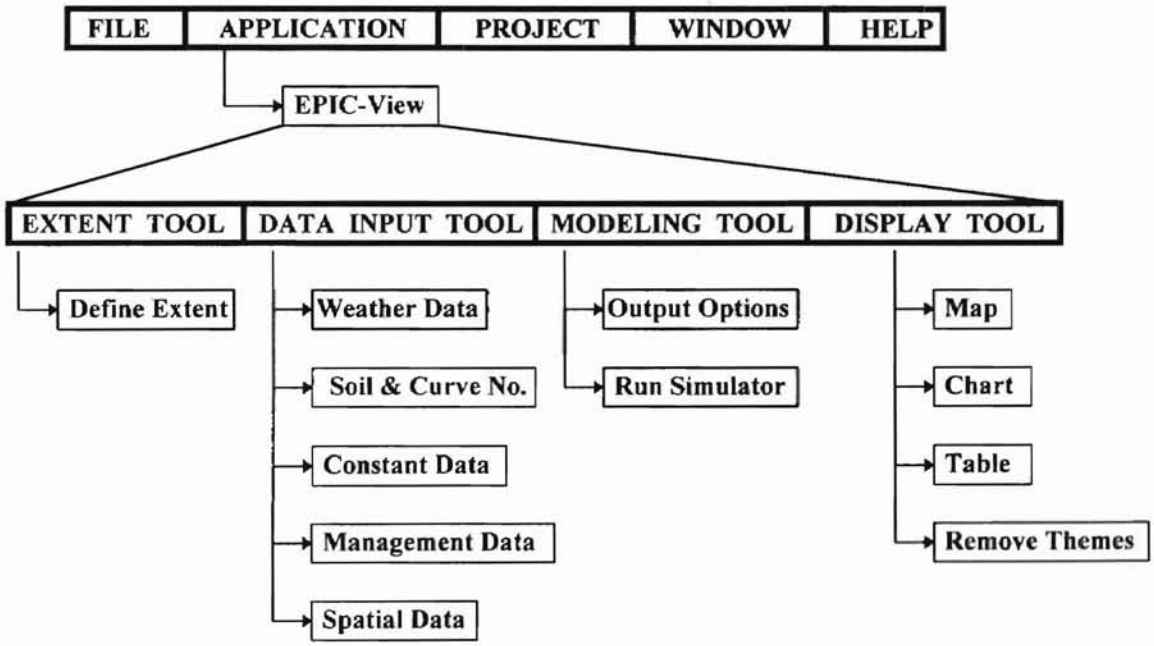


Figure 1. Main Menu for EPIC-View.

Various data variables identified from EPIC input dataset are classified, as per different tools shown above (as shown in Appendix D).

3.2 Description of Various Menu Options

1. Extent Tool

This menu has following option:

Define Extent:

This menu option opens the field view designated by a user and activates the main theme as shown in Appendix C.17.

2. Data Input Tool

This menu provides various options required to create the EPIC input dataset. It consists of following options:

Weather Data:

This option displays a Visual Basic screen which provides a user with two choices, whether a user has weather files for the field or EPIC generated weather file is to be loaded. In former case, a user has to enter the weather file name with complete path and in the later case, a user has to enter the latitudes and longitudes of the field, and EPIC will automatically load the weather file from the weather station nearest to the user's field. As a result a file "const.utl" is created and a line is written to it, depending upon the user's choice, e.g., "@<path><file name>" or "LOCWEAT <latitude> <longitude>" for former and latter cases respectively. The Weather data screen is shown in Appendix C.1. The data flow diagram is shown in Appendix B.1. A typical "const.utl" file is shown in Appendix E.4.

Soil Data:

This option is enabled if a user does not have field specific soil data. In that case, a soil from this option can be selected which becomes generic for the whole field. This option displays a Visual Basic screen which provides a user with a list of soils, provided by EPIC. A user can also select the Run off Curve Number based upon the soil selected. The Soil data screen is shown in Appendix C.2. The data flow diagram is shown in Appendix B.2. Lines, e.g., "GETSOIL <soil code>" and "CN2 <curve number>" are written to "const.utl". A typical "const.utl" file is shown in Appendix E.4.

Constant Data:

Data that remains constant for the entire field is entered (at one time only) and replicated for all the existing cells in the gridded field. The constant data option is enabled only when a user has entered weather data (and soil data). This menu options presents a Visual Basic screen which displays default values of all the variables, required by EPIC, which remain constant. A user can make required modifications and save these variables in a constant dataset which is replicated for all the cells in the grid. Various lines, one for each constant variable, are written to "const.utl" and then a constant data set, e.g., "const.dat" is created from "const.utl" using UTIL. Then, this constant dataset is replicated for all the cells in the grid. The Constant data screen is shown in Appendix C.3 & C.4. The data flow diagram is shown in Appendix B.3. A typical "const.utl" file is shown in Appendix E.4.

Management Data:

This tool allows a user to enter management practices, carried out in the field. A user has choice to make a set of management practices generic for the whole field, in that case, all the management practices, a user enters, become applicable to all the cells in the gridded field. In other case, a user can select a set of cells and then select this tool to enter these cells specific management practices, which are applicable to all selected cells. A Visual Basic screen is presented with a list of EPIC supported management practices along with a list of all months and days to select the date of operation. Upon selecting a management operation, a user is presented with another screen which displays further choices to be made, e.g., upon choosing "FERTILIZE" as management operation, a user is presented with another screen showing a list of fertilizers to choose from, fertilizer application rate, etc. (as shown in Appendix C.8). Upon choosing "OK", the operation is written to cell specific files, e.g., "mgmtCellId#.utl", for all selected cells. These files will be loaded to the cells' input data set at the time of running EPIC on these cells. . The Management data screen is shown in Appendix C.5. The data flow diagram is shown in Appendix B.4. A typical "mgmtCellId#.utl" file is shown in Appendix E.2.

Spatial Data:

This option extracts the spatial attributes from all selected cells, selected by a user to run EPIC. Spatial attributes, such as soil, crop, elevation, area, slope, etc. are stored in cell specific files, e.g., "formCellId#.utl", which are loaded when running the EPIC

model on individual selected cells. The data flow diagram is shown in Appendix B.5. A typical "formCellId#.utl" is shown in Appendix E.3.

3. Modeling Tool

The modeling tool is also an interactive tool. This consists of two following options :

Output Options Tool:

This options allows a user to specify the variables which are desired to be monitored as the output from EPIC. A Visual Basic screen is displayed with a list of EPIC supported output variables. A user can select more than one variables. EPIC sets a limit of 30 variables which can be chosen as output. A user also has a choice of selecting daily, monthly, yearly, annual, or all output files options which allows EPIC to generate these files while running on individual cells' input datasets. These variables are then written to a "prnt.utl" file which is later loaded to the "prnt5300.dat" file at the time of running EPIC of selected cells. The data flow diagram is shown in Appendix B.6 and the Output options screen is shown in Appendix C.15. A typical "prnt.utl" file is shown in Appendix E.5.

Run Simulator:

On selecting this tool, a Visual Basic waitshell is executed to run UTIL and EPIC model on each individual cell's input datafile. Then, the corresponding output files are parsed to create a comma delimited file which can be retrieved in ArcView as a table. The dataflow diagram is shown in Appendix B.7.

4. Display Tool

This menu remains disabled until Run Simulator is invoked which runs EPIC on the selected cells. It consists of following options:

Map:

This option adds the parsed output (from EPIC) as a table and joins it with the attribute table of the main theme. New themes are created corresponding to each variable in the parsed table. The variables' values are shown in the corresponding themes in the form of equal intervals. Thus a user can monitor the effects on each variable, as analyzed by EPIC, in a spatial manner and can make better comparison. The screen after displaying results is shown in Appendix C.18.

Table:

This option displays the parsed table, added to ArcView. A user can monitor values of different variables visually in a tabular form. A typical table is shown in Appendix C.19.

Chart:

This option displays a selected variable, from the list of variables in parsed table, in the form of percentage of values falling in each of the equal intervals. This provides a comprehensive summary in terms of whether the values fall within allowable ranges or not. A typical chart is shown in Appendix C.20.

Remove Themes:

This option allows a user to remove all added themes, as a result of EPIC run. The main theme remains active.

3.3 Weather Data Tool

Weather data tool provides a user with a screen which provides two choices. A user can either choose to user EPIC provided weather file by entering the latitudes and longitudes of the field or if a specific weather file is available, its path can be specified, which will be loaded to the constant dataset. If former option is chosen, EPIC loads the weather file from a weather station nearest to the field. This information is stored in a file "const.utl". The data flow diagram is shown in Appendix B.1.

3.4 Soil Data Tool

This option is enabled only if a user does not have his own soil file. This tool presents a screen as shown in Appendix C.2 where a user has a choice of soil and run off curve number. The selected soil becomes generic for the whole field. This information is appended in a file "const.utl". The list of soils is created from a file **soil.lst**. The data flow diagram is shown in Appendix B.2.

3.5 Constant Data Tool

Constant data tool allows a user to modify various variables which remain constant for the whole field. These variables are loaded into a constant dataset which is then replicated for all cells in the gridded coverage. The tool presents a user with a screen as shown in Appendix C.16. Various variables' values are written out into a file "const.utl" which is then loaded to constant dataset "const.dat" using the UTIL. The data flow diagram is shown in Appendix B.3.

3.6 Management Practices Tool

Management practices are entered as soon as management zone is delineated. A user is prompted with a Visual Basic screen as shown in Appendix C.5 to enter various operations and also to delineate more management zones if he chooses to. The data flow diagram is shown in Appendix B.4.

The user can select the month and date of operation. A management operation can be selected from a list of operations, provided, which is available in a file **Mgmtoper.dat**. Month and date are stored in EPIC input variables MON and DAY respectively. Operation code, corresponding to the operation, the user selects, is automatically looked up from the file and stored in the variable COD. Depending upon the operation code a different Visual Basic screen pops up as shown in Appendix C.6-C.15.

Here, e.g. if a user chose **Fertilize** as the operation, the screen shown in Appendix C.8 presents a list of fertilizers, available in the file **Fertdata.dat**. A user can select a fertilizer, the corresponding fertilizer Id# is stored in variable FN. The user needs to enter values for Fertilizer Application rate(FAP), Heat Unit scheduling(HUSC). These values are then loaded in all the cells' input datasets, which fall under this management zone.

Similarly, the user can enter another operation and so on. In case a crop rotation occurred, the user can enter new set of management practices, thus can have different sets of management practices for different crops. The user can also delineate a new management zone, by marking out a new zone on the field view, and enter management practices for it in a similar way. A user can also make one set of management practices as

generic, i.e. they apply to the whole field. In this case, all cells' input dataset have common management practices. Management related variables are provided to a user to be modified if chosen to, as shown in Appendix C.15.

There is an option to view a summary of management operations entered, which provides a list of all operations entered, per management zone.

Following lookup files are required for management practices operations:

Usdacrop.txt - for crop selection and extracting crop code.

Usdapest.dat - for pesticide selection and extracting pesticide code.

Mgmtoper.dat - for management operation selection and extracting operation code.

Fertdata.dat - for fertilizer selection and extracting fertilizer code.

3.7 Spatial Data Tool

This tool extracts various attributes from the field's main theme table. The attributes are crop, elevation, slope, soil, run off curve number, watershed area, etc. These attributes are stored in the cell specific files (form#.utl, # - cell Id.). These will later be loaded in the cell specific datasets at the time of running EPIC on selected cells. The data flow diagram is shown in Appendix B.5.

3.8 Output Options Tool

This tool presents a screen where a user can choose the variables desired to be monitored as a result of running EPIC on selected cells. The EPIC output will be displayed in terms of these variables. A user can choose at most 30 variables as a limit provided by EPIC. A user can also select so that EPIC generates daily, monthly, yearly,

annual, or all these output files. These options, selected by a user are stored in a file "prnt.utl" which are later loaded into "prnt5300.dat" at the time of running EPIC on selected cells. The list of output variables is created from a file **opvarlst.dat**. The data flow diagram is shown in Appendix B.6.

3.9 Run Simulator Tool

This menu option creates various batch files which are needed to complete the cell specific input datasets and also invoking EPIC on all these datasets and finally parsing the EPIC output into a comma delimited file. Various files created are "create.bat" (shown in Appendix E.6) and "runepic.bat" (shown in Appendix E.7). The data flow diagram is shown in Appendix B.7.

3.10 Map

This menu option in the Display Tool, allows a user to load the parsed file as a table and join it to the field theme's attribute table and also create new themes, for each output variable selected by a user, which are then added to the main field view providing a powerful visual representation of the model run on the selected cells. As shown in Appendix C.18.

3.11 Chart

This menu option in the Display Tool, allows a user to display a chart in the form of a bar graph or a pie chart for output variables selected by a user. . As shown in Appendix C.20.

3.12 Table

This menu option in the Display Tool, allows a user to display the added table created as a result of parsing EPIC output. . As shown in Appendix C.19.

3.13 Remove Themes

This menu option in the Display Tool, allows a user to remove all added themes as a result of choosing the Map menu option from Display Tool. As a result only the main theme remains in the field view.

CHAPTER IV

CONCLUSIONS AND RECOMMENDATIONS

This work has dealt with the design and implementation of an interactive system which integrates ArcView with EPIC model. The interface developed automates the task of entering management practices for a specific part of field and also in creation of input datasets for running EPIC model. EPIC is then invoked on the specified part of the field and the results are parsed into ArcView readable format so that they can be loaded in the form of table and displayed spatially on the field view to provide recommendations to a user. Following are specific conclusions and recommendations which can be made based on this work.

4.1 Conclusions

1. EPIC-View was developed by making use of ArcView's development environment language Avenue and the user interface data entry screens were developed in Visual Basic.

2. EPIC-View is capable of allowing a user to enter management practices for different management zones at a time. Also, a user can overlap different management zones to enter different management practices, i.e., a user can select some zones (cells) and enter a set of management practices and then selects some other zones and enter different management practices, so that some zones(cells) are common and receive both the sets of management practices.

3. EPIC-View was successfully able to create input datasets for each cell (management zone) and able to run EPIC on the selected cells. This process eliminates the need to manually create input datasets for each management zone and run the EPIC model on these datasets and then pull the results back to ArcView.

4. The results generated were tested to be correct and provided visual representation of the result of different management practices carried out on the selected zones.

4.2 Recommendations for Future Research

1. The interface developed assumes that the field view available to it is already gridded. An addition to it could be to allow a user to open his field view and grid the field using the gridding tool which will be available in the new version of ArcView.

2. The display tool can be made more powerful to be able to display various comparison results after multiple runs of the model.

3. EPIC's daily, monthly, yearly, annual outputs can also be made use of in showing the results spatially. These files are generated with extensions .epd, .epm, .epy and .epa respectively.

4. Capability to store the results of different runs could be incorporated so that comparisons between different runs could be made.

5. In the Management Practices Tool can be enhanced so that it is able to summarize all the management practices entered so far and also allows a user to modify already entered management practices.

BIBLIOGRAPHY

ArcView, Version 2.1(ESRI) User's Manual.

Arnold, J.G., Engel, B.A., and Srinivasan, R. (1993). Continuous time, grid cell watershed model. Application of Advanced Information Technologies: Effective Management of Natural Resources. ASAE Publication 04-93. pp. 267-278.

Arnold, J.G., Williams, J.R., Nicks, A.D., and Sammons, N.B. (1990). Basin scale simulation model for soil and water resources management. College Station, TX: Texas A&M Univ. Press.

Environmental Systems Research Institute, Inc. (1994a). Avenue - Customization and Application Development for ArcView.

Environmental Systems Research Institute, Inc. (1994b). ArcView - The Geographic Information System for Everyone.

Jankowski, P., and Haddock, G. (1993). Integrated nonpoint -source pollution modelling system. *In* Second International Conference/Workshop on integrating GIS and Environmental Modeling. Breckenridge, Colorado. 26-30 Sept. 1993.

Jones, J.W., and Ritchie, J.T. (1991). Crop growth models. *In* G.J. Hoffman, T.A. Howell, and K.H. Solomon (Eds.). Management of Farm Irrigation Systems. Amer. Soc. Agricultural Engineering, St. Joseph, MI. pp. 63-89.

Knisel, W.G., Jr. (1980). "A field Scale Model for Chemicals, Runoff and Erosion from Agricultural Management Systems." Cons. Res. Rpt. No. 26. USDA. 640.

Onstad, C.A., and Foster, G.R. (1975). Erosion modeling on a watershed. TRANSACTIONS of the ASAE 18(2):288-292.

Rewerts, C.C., and Engel, B.A. (1991). ANSWERS on GRASS: Integrating a watershed simulation with a GIS. ASAE Paper No. 91-2621.

Richardson, C.W. (1982b). A wind simulation model for wind erosion estimation. ASAE Paper No. 82-2576, ASAE, St. Joseph, MI 49085.

Shanholtz, V.O., Desai, C.J., Zhang, N., Kleene, J.W., Metz, C.D., and Flagg, J.M. (1990). Hydrologic/water quality modeling in a GIS environment. ASAE Paper No. ()-3033. ASAE, St. Joseph, MI.

Sharpley, A. N., and Williams, J.R. (Eds). (1990). EPIC - Erosion/Productivity Impact Calculator. Model documentation. U.S. Department of Agriculture Technical Bulletin No. 1768.

The U.S. Department of Agriculture Agricultural Research Service at Temple, Texas EPIC User's Guide - Draft.

Visual Basic, Version 4.0 (Microsoft Corp.) User's Manual.

Williams, J.R., Dyke, P.T., and Jones, C.A. (1983). EPIC- A model for assessing the effects of erosion on soil productivity. Proceedings Third International Conference on State-of-Art in Ecological Modelling, Colarado State University, May 24-28, 1982.

Williams, J.R., Jones, C.A., and Dyke, P.T. (1984). A Modeling Approach to Determining the Relationship Between Erosion and Soil Productivity. Transactions of the ASAE.

Yoon, J., Padmanabhan, G., and Woodbury, L.H. (1993). Linking agricultural nonpoint pollution model (AGNPS) to a geographic information system (GIS). Proceedings of the symposium on Geographic Information Systems and Water Resources. AWRA. pp. 79-87.

APPENDICES

APPENDIX A

USING THE SYSTEM

EPIC-View can be used by opening the project file under ArcView 2.1. The project file is named under **EPICView.apr**. Initially at the time of installation, a multiple input entry screen is displayed where a user can enter different directory paths as shown in Appendix C.21. These paths are written to a file **EVPaths.txt** which is created under C:\. Here it is assumed that a user has already created his field view which has overlaid layers of spatial attributes in the form of a grid. Path to this view polygon needs to be specified here.

Subsequent openings of this project file will read various paths from this file and thus the multiple entry screen is not displayed every time. Once, the project is opened, a user can view his gridded field by selecting **Define Extent** menu option. This option opens the field view as shown in Appendix C.17. Now a user can select the **weather data** (described in Section 3.3) menu option to specify his weather file or enter the latitudes and longitudes of the field. If a user had specified soil and runoff curve number in the field view's main theme's attribute table, the soil and curve number is directly extracted while running **Spatial Data** tool (described in Section 3.7) otherwise **Soil Data** (described in Section 3.4) menu option needs to be selected where a list of EPIC supported soils is presented and run off curve number can be selected. Now constant data set for the whole field can be created by selecting **Constant Data** tool (described in Section 3.5), where a user can modify variables that remain constant for the whole field. These variables will be loaded to a constant dataset which will be replicated for all the cells in the gridded field. Now a user can select **Management Data** tool (described in

Section 3.6) where he can enter management practices for the field either by making one set of management practices generic for the whole field or by selecting an area of field and entering management practices for the same. The Management Data tool provides a screen where a user can select management operations from a list and enter them in cell specific files. All the above operations are done one time only. Ofcourse, management practices might be entered on a regular basis but constant data tool need not be invoked until a user wishes to change some of the constant variables' values. Now to run the model, a user needs to select a portion of the field by highlighting those cells, either by using the highlighting tool button or using query builder. The highlighted cells are displayed by a yellow color (Shown in Appendix C.17). Now a user can select the **Spatial Data** tool (described in Section 3.7), which extracts the spatial attributes from the main attribute table for all selected cells. Now a user can select the **Output Options** tool (described in Section 3.8) to select the output variables he is interested to monitor after running EPIC. After that, the **Run Simulator** menu option (described in Section 3.9) can be invoked which executes batch files and invokes UTIL, EPIC and finally the Parser to parse the output generated by EPIC into a comma delimited file.

After the user returns back to ArcView, he can select the **Map** (described in Section 3.10) menu option from Display tool and view the results of model run, spatially. This is done by adding more themes for each output variable, a user selected, into the main field view as shown in Appendix C.18. A user can also generate charts by selecting **Chart** from Display tool as shown in Appendix C.20 and also view the results table **form.prs** by selecting **Table** from Display tool as shown in Appendix

C.19. A user can remove the newly added themes by selecting the **Remove Themes** menu option which deletes all but the main theme. Similarly, a user can select a different portion of field view and run model on it and monitor the results.

Various files needed to run the model are as follows:

EPICView.apr -- The main ArcView project file.

Constdat.exe -- The constant data user interface.
Dsetmake.exe -- The model running wait shell.
Outputop.exe -- The output options user interface.
Mgmt.exe -- The management practices user interface.
Parse.exe -- The parser.
Weather.exe -- The weather data user interface.
Soil.exe -- The soil data user interface.

Curvenum.dat -- The data file containing a list of runoff curve numbers.
Fertdata.dat -- The data file containing a list of fertilizers.
Mgmtoper.dat -- The data file containing a list of management operations.
Opvarlst.dat -- The data file containing a list of output variables.

EPIC5300 model files.

Recommended directories:

Create a directory **EPICView** under C:\. Create a directory **EPIC5300** under C:\ to store the EPIC5300 files. Create following directories under C:\EPICView :

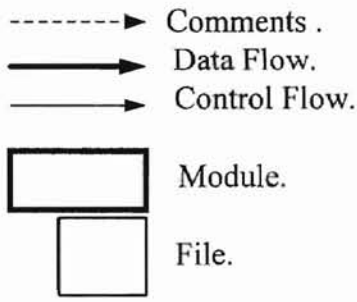
Project -- To keep the EPICView.apr project file.
Temp -- To keep all the created files.
Weather -- To keep weather file if a user has his own weather file (with .utl extension).
Soil -- To keep soil files if a user has his own soil files (with .utl extension).
EXEDir -- To keep all the executable files described above.
Field -- To keep the files of the field view including the database table.

It is recommended that a user makes a copy of the EPICView.apr file and user the copy so that the original project file is not altered.

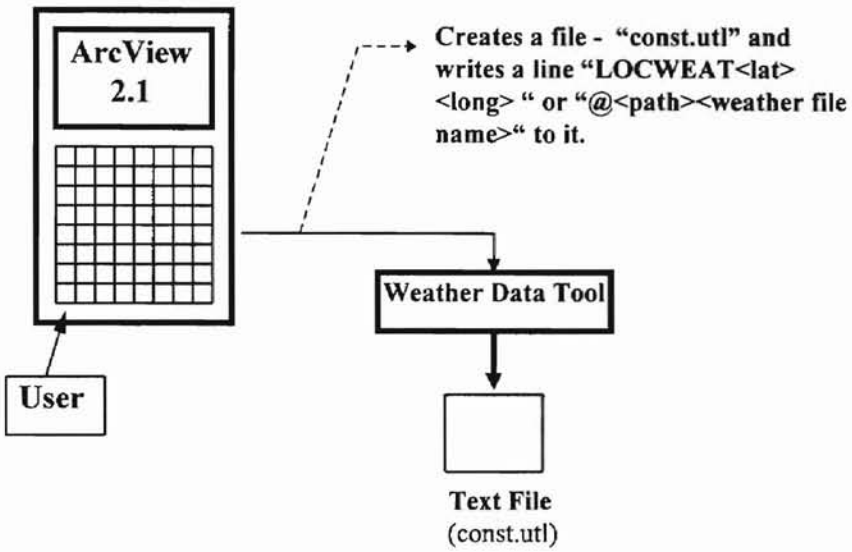
APPENDIX B

DATA FLOW DIAGRAMS

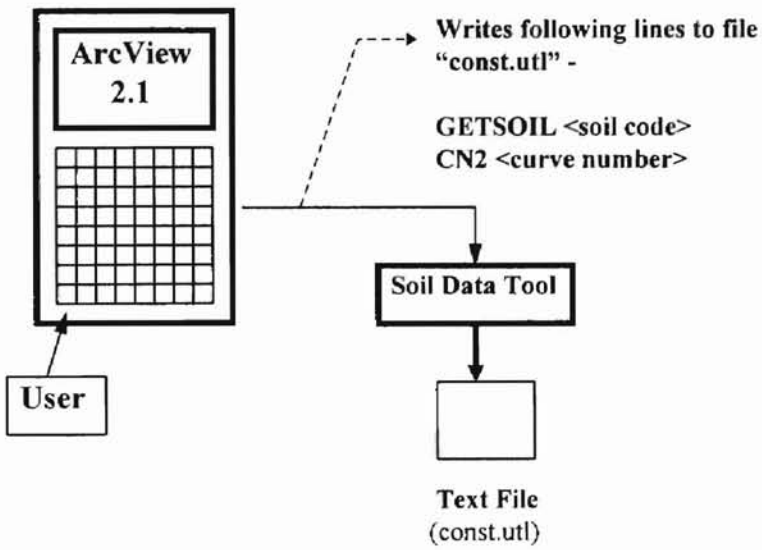
LEGEND:



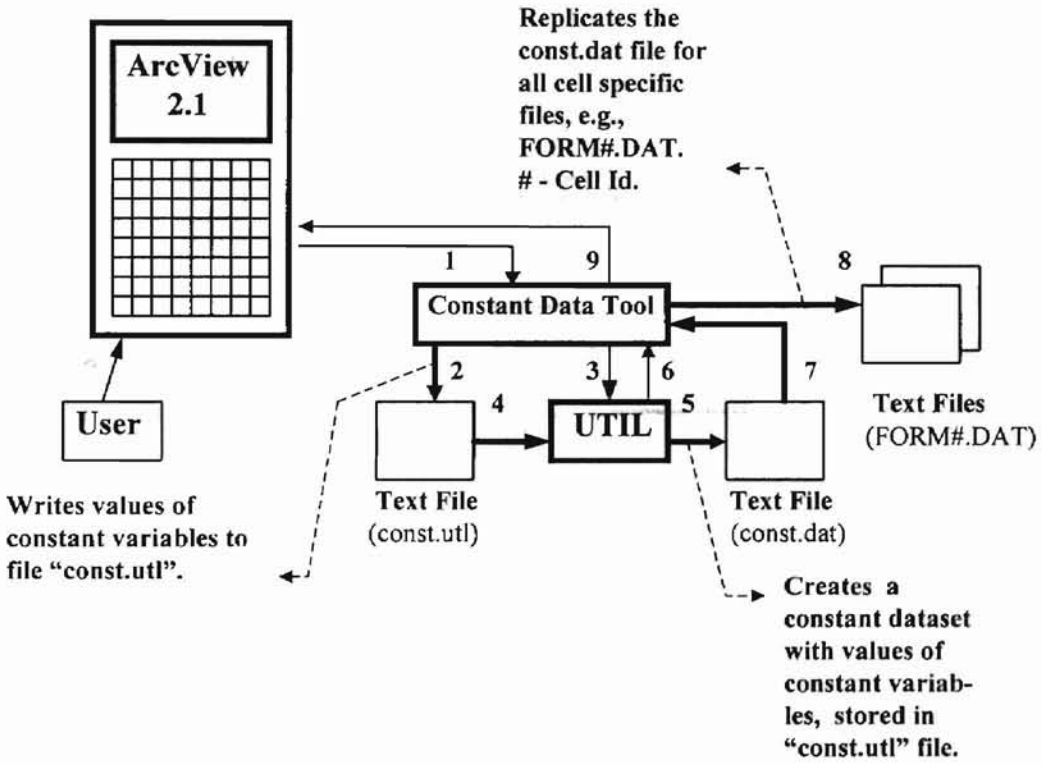
-- Cell Id Number.
Specified Tools -- Data entry screens.



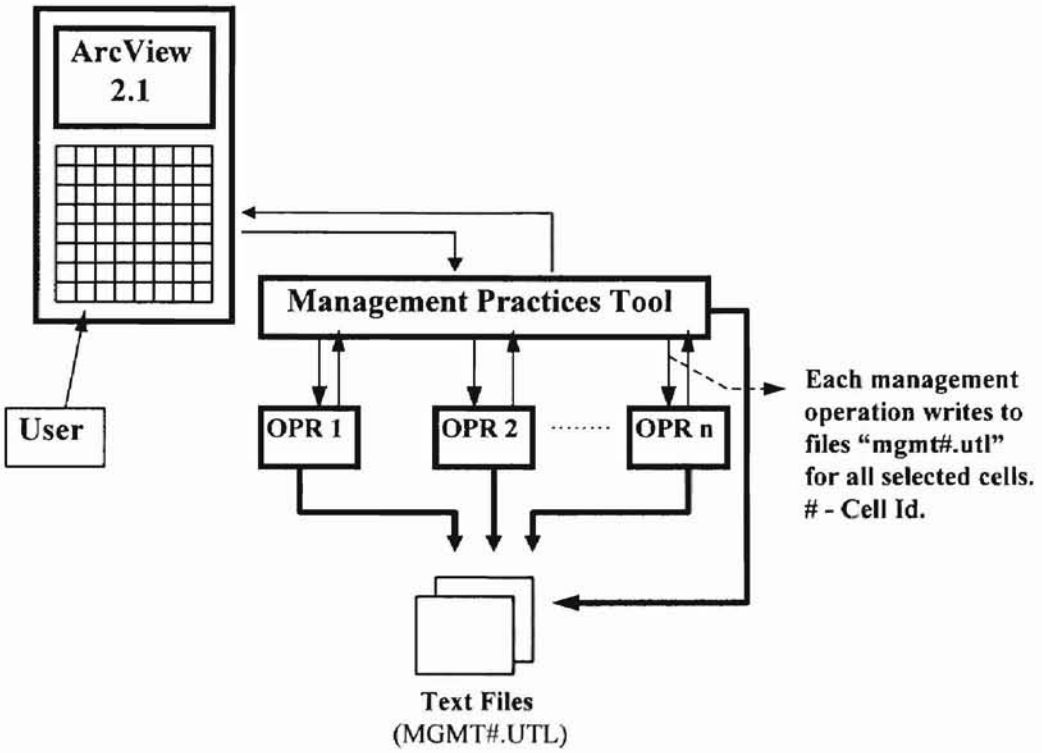
B.1 Data Flow Diagram for Weather Data Menu Option.



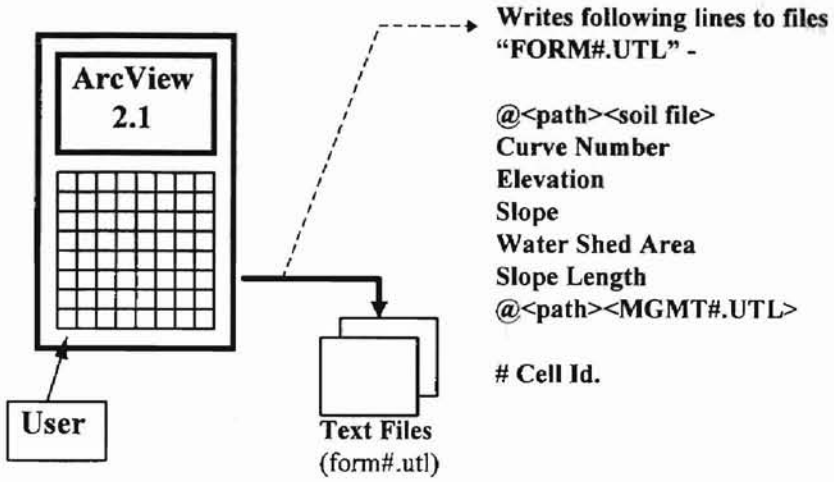
B.2 Data Flow Diagram for Soil Data Menu Option.



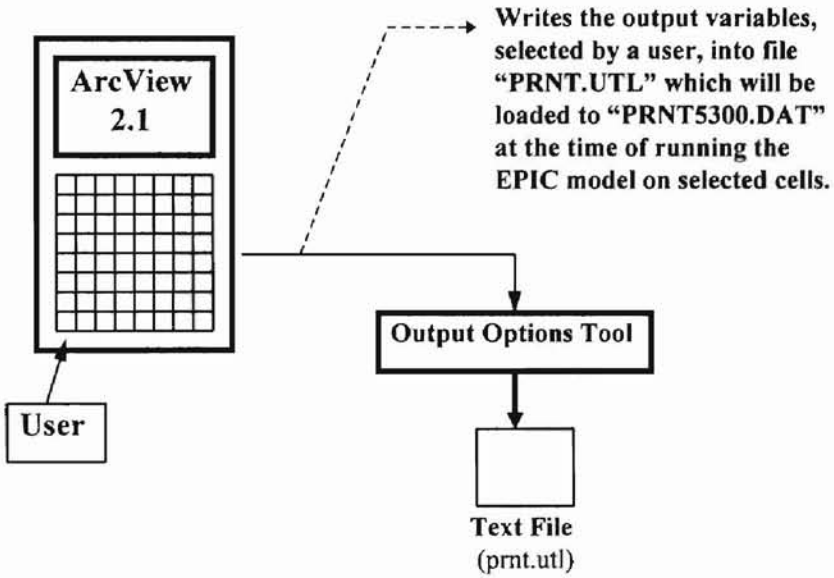
B.3 Data Flow Diagram for Constant Data Menu Option.



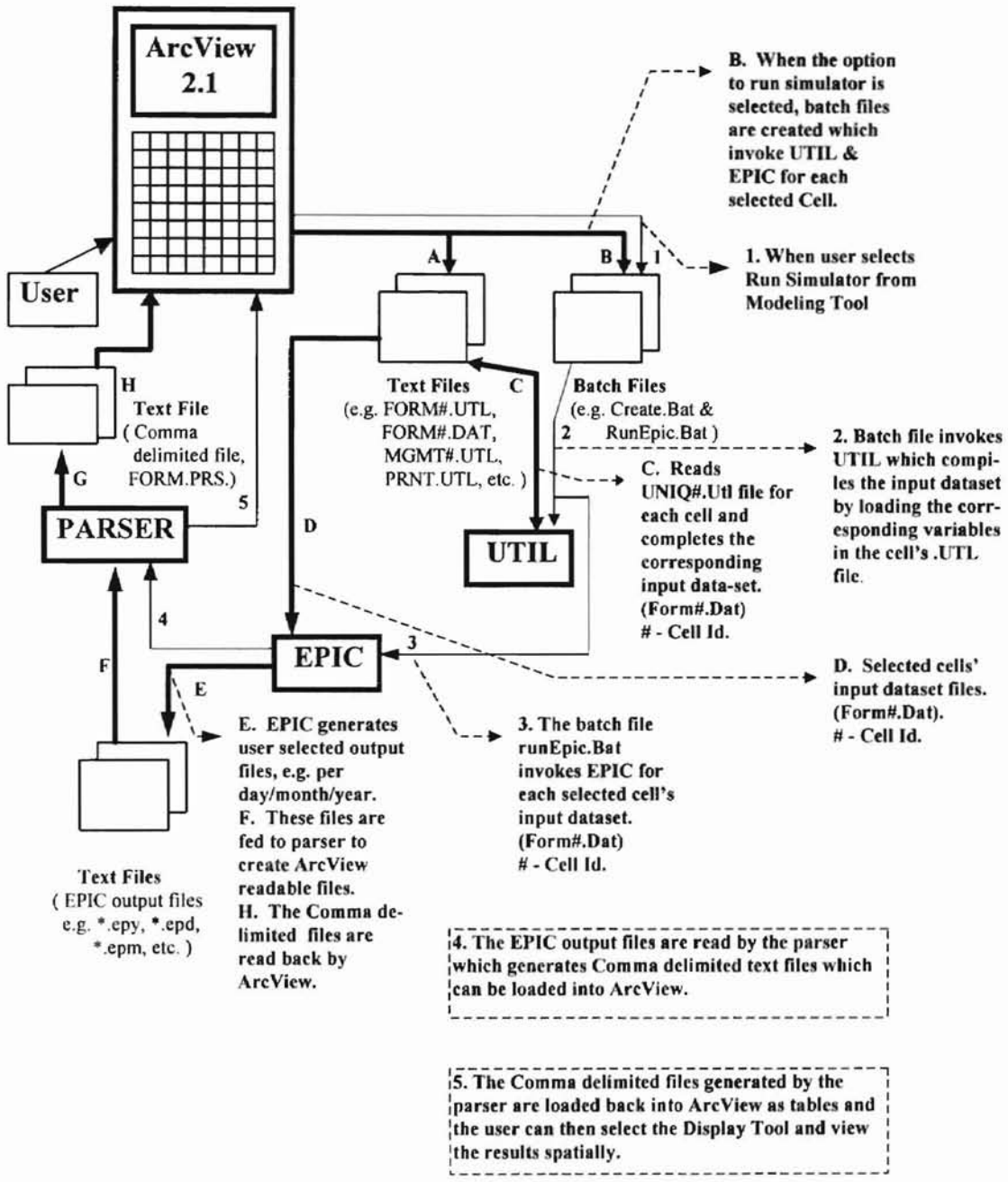
B.4 Data Flow Diagram for Management Practices Menu Option.



B.5 Data Flow Diagram for Spatial Data Menu Option.



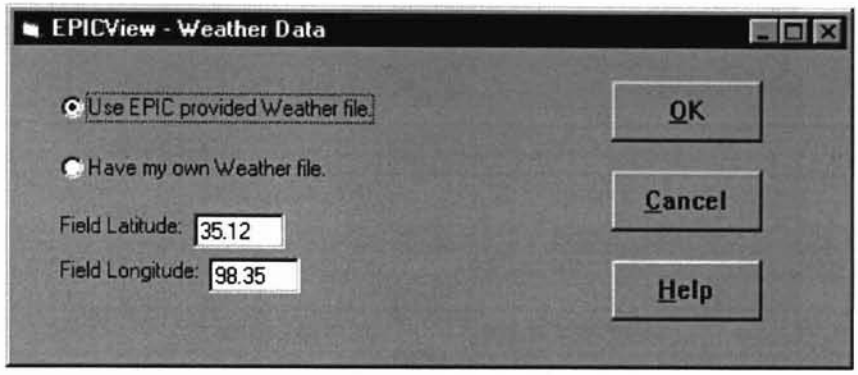
B.6 Data Flow Diagram for Output Options Menu Option.



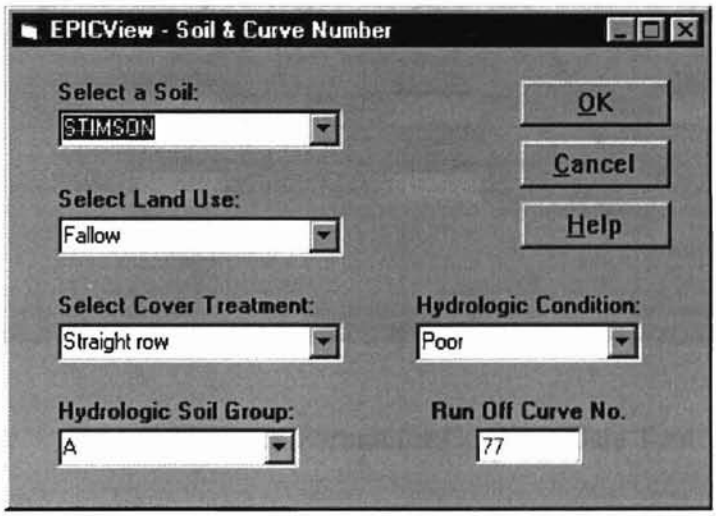
B.7 Data Flow Diagram for Run Simulator Menu Option.

APPENDIX C

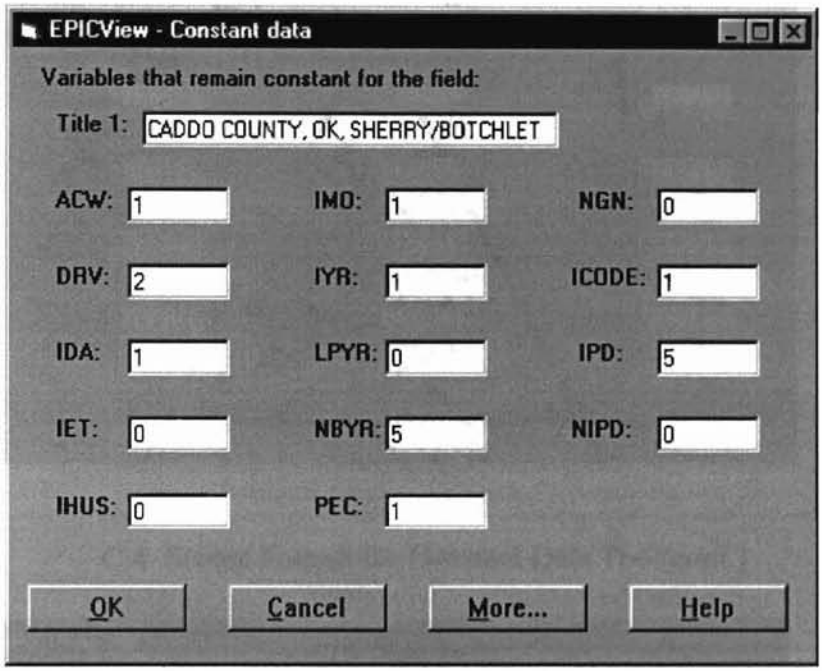
SCREEN FORMATS



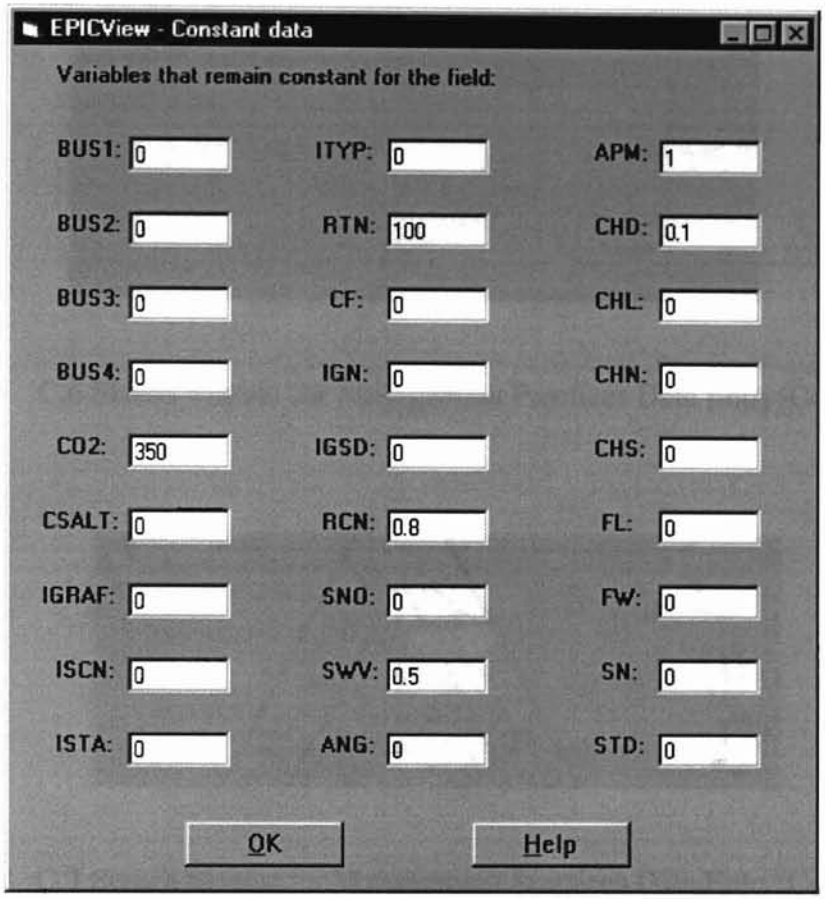
C.1 Screen Format for Weather Data Entry.



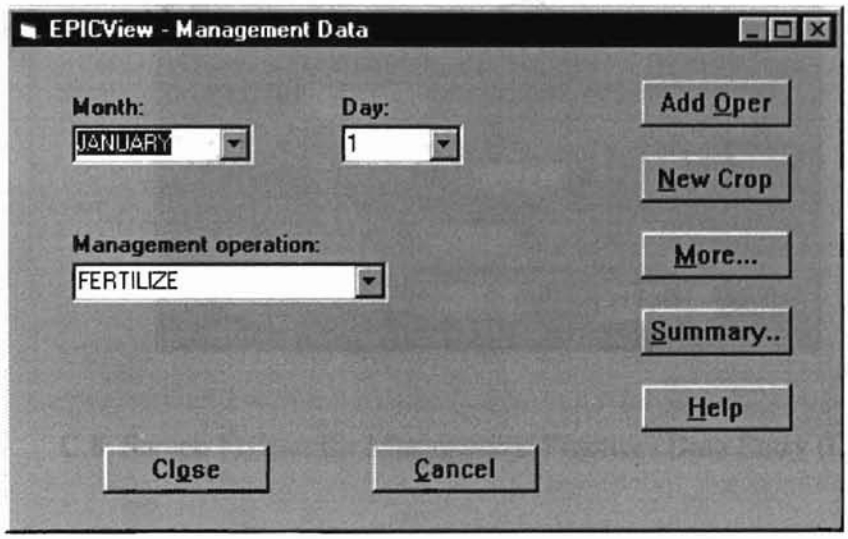
C.2 Screen Format for Soil & Curve Number Data Entry.



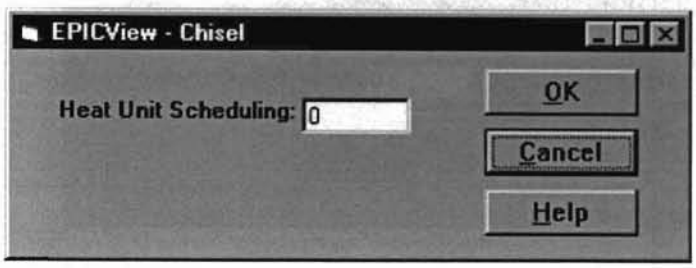
C.3 Screen Format for Constant Data Tool.



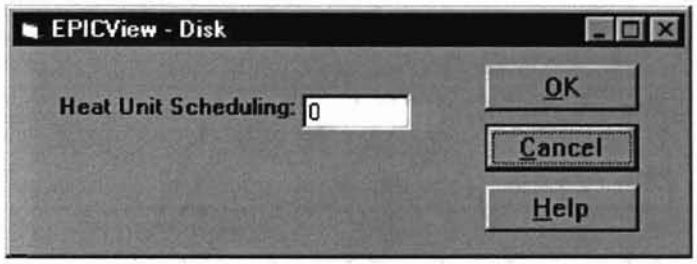
C.4 Screen Format for Constant Data Tool(cont.).



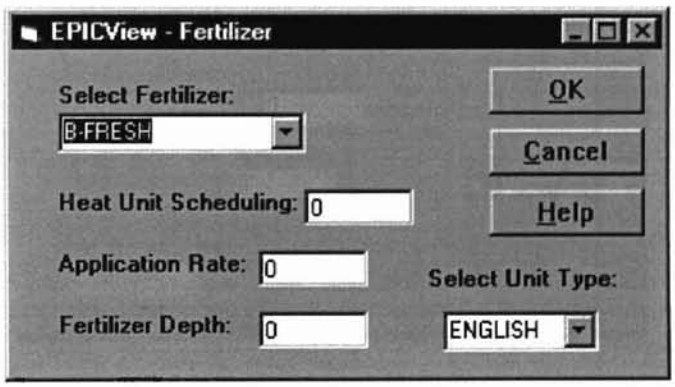
C.5 Screen Format for Management Practices Data Entry.



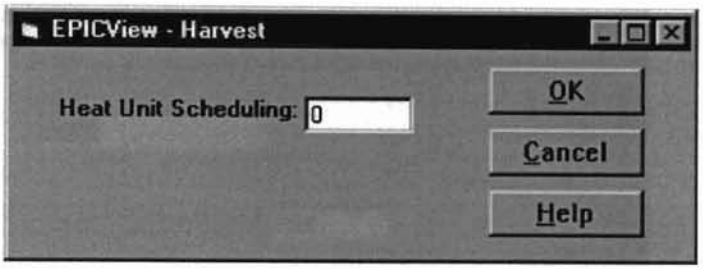
C.6 Screen Format for Management Practices Data Entry(Cont.).



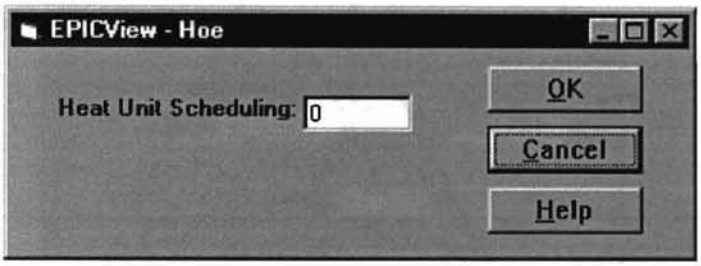
C.7 Screen Format for Management Practices Data Entry(Cont.).



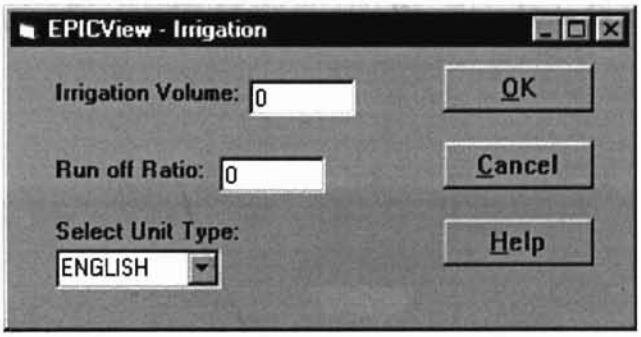
C.8 Screen Format for Management Practices Data Entry (Cont.).



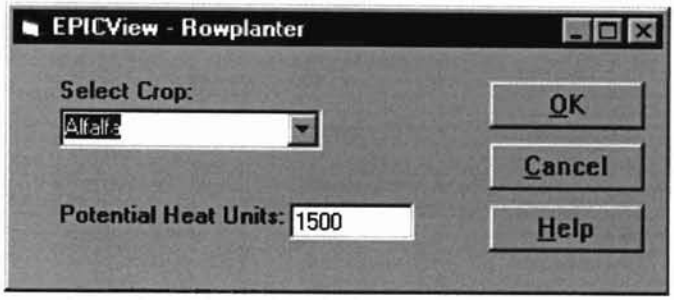
C.9 Screen Format for Management Practices Data Entry (Cont.).



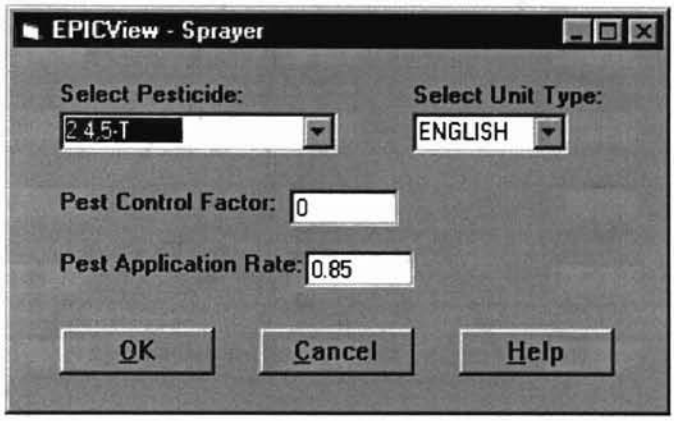
C.10 Screen Format for Management Practices Data Entry (Cont.).



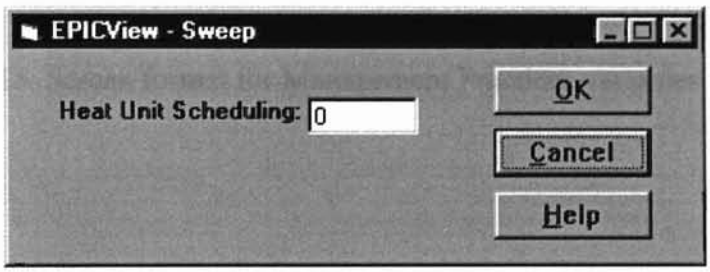
C.11 Screen Format for Management Practices Data Entry (Cont.).



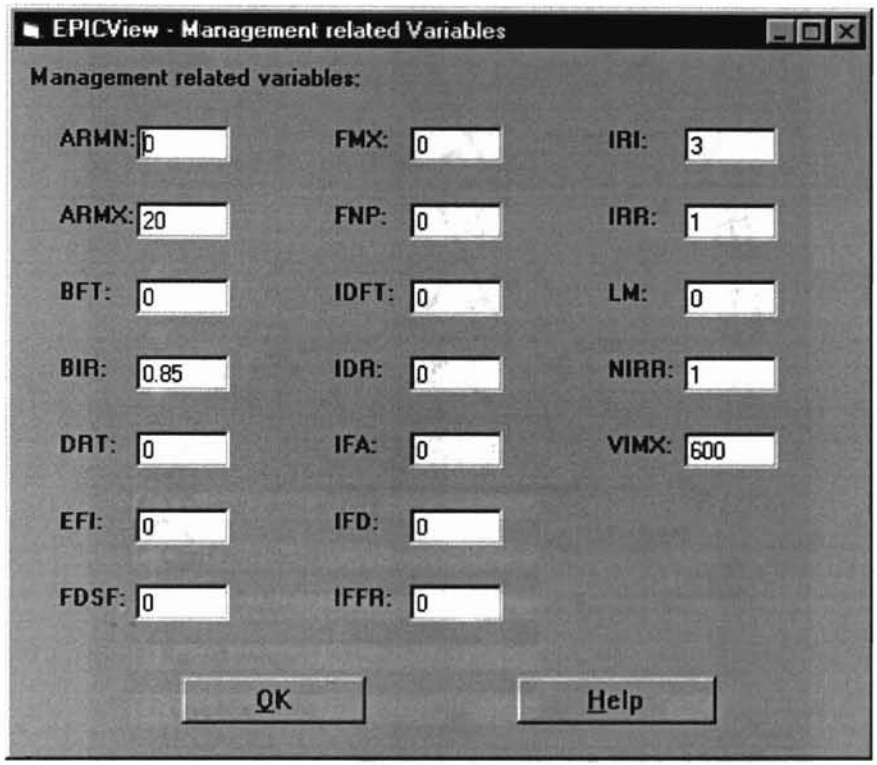
C.12 Screen Format for Management Practices Data Entry (Cont.).



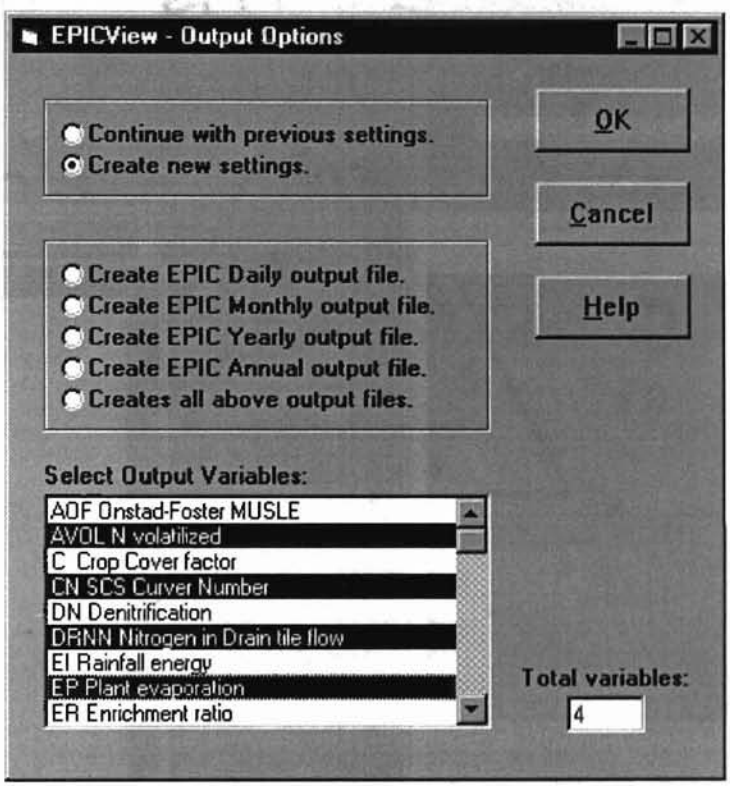
C.13 Screen Format for Management Practices Data Entry (Cont.).



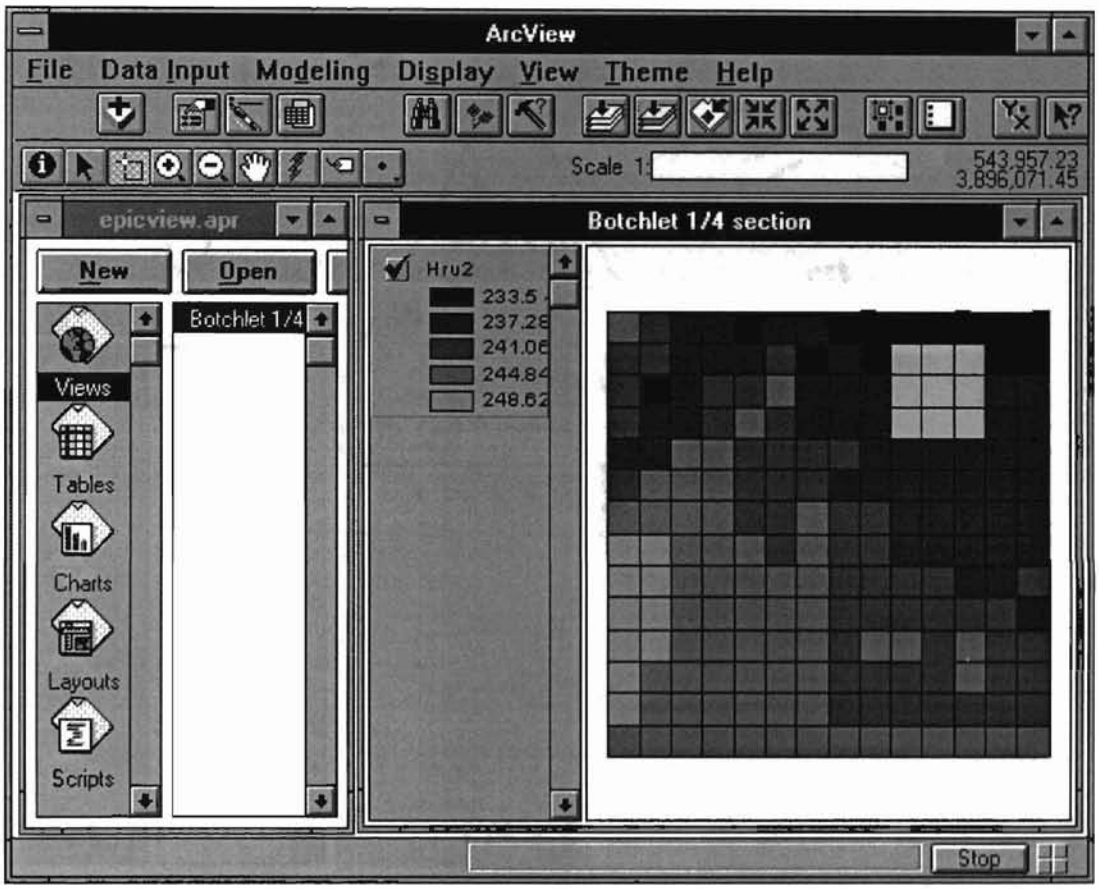
C.14 Screen Format for Management Practices (Cont.).



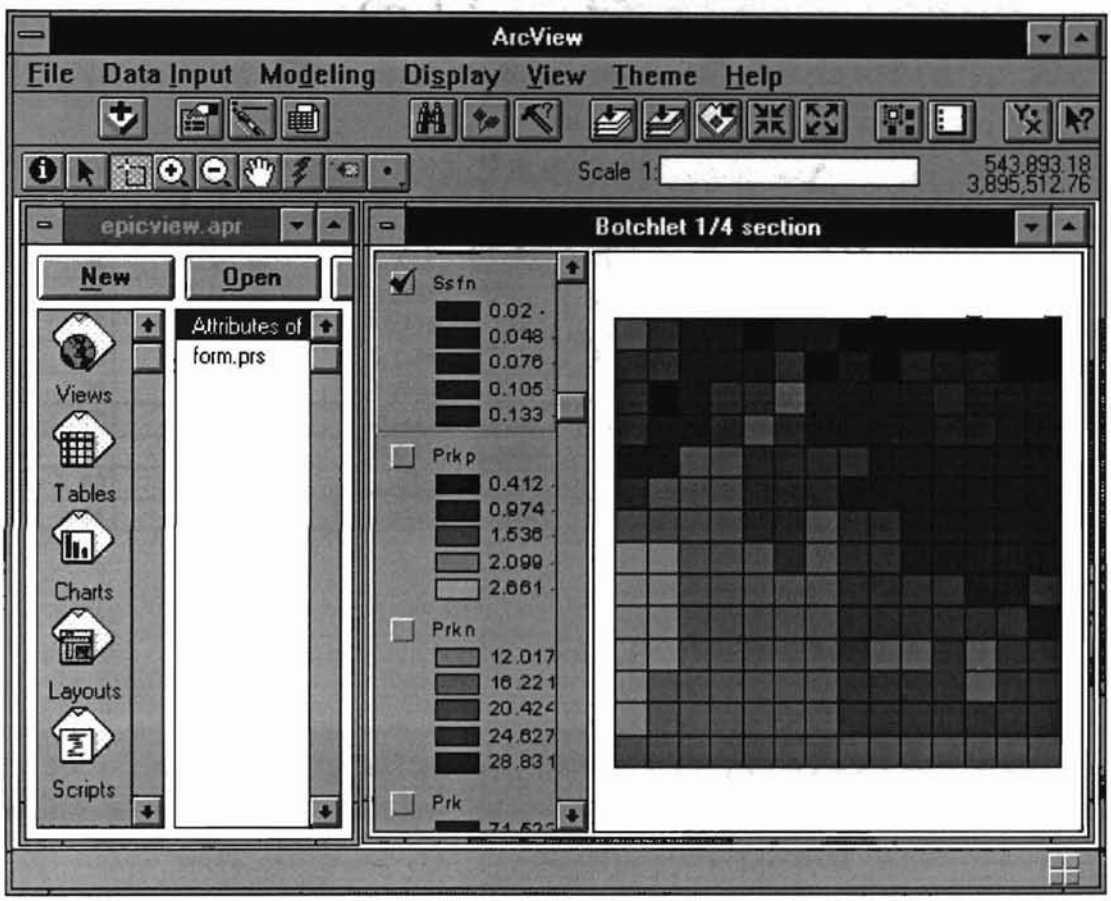
C.15 Screen format for Management Practices variables.



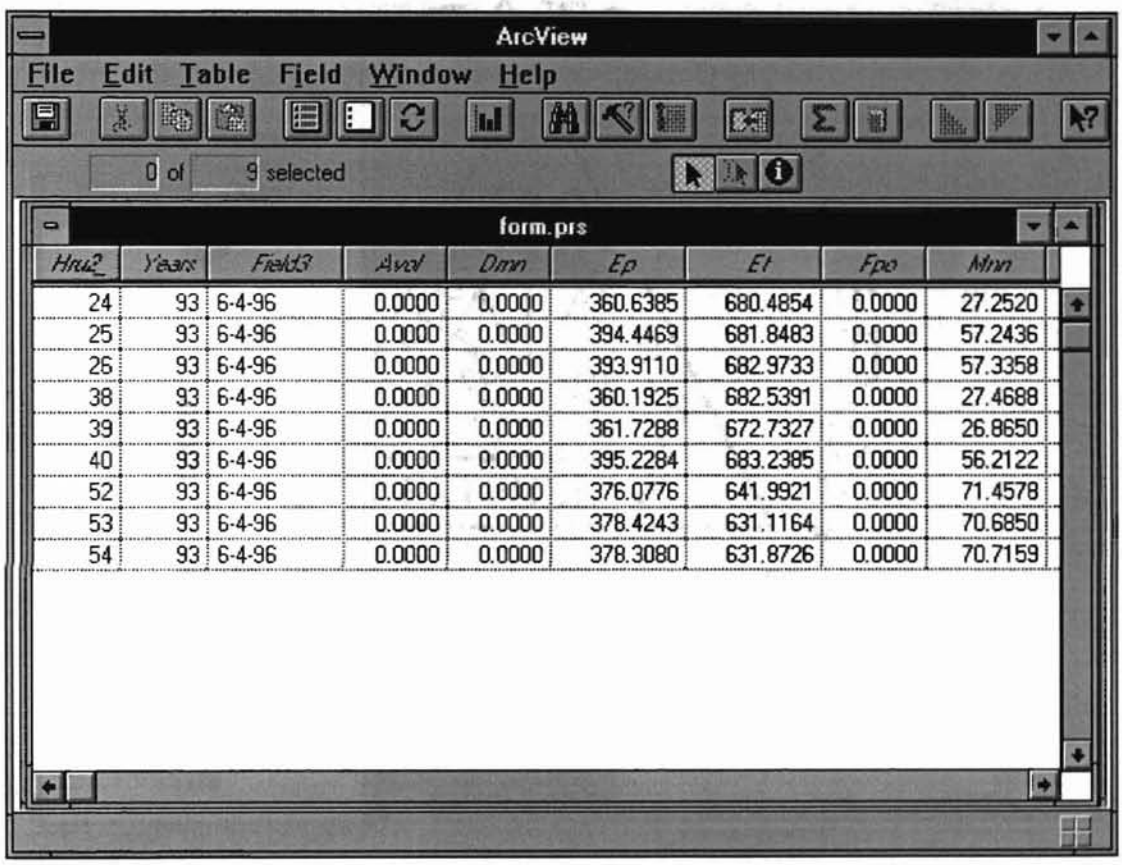
C.16 Screen Format for Output Options Data Entry.



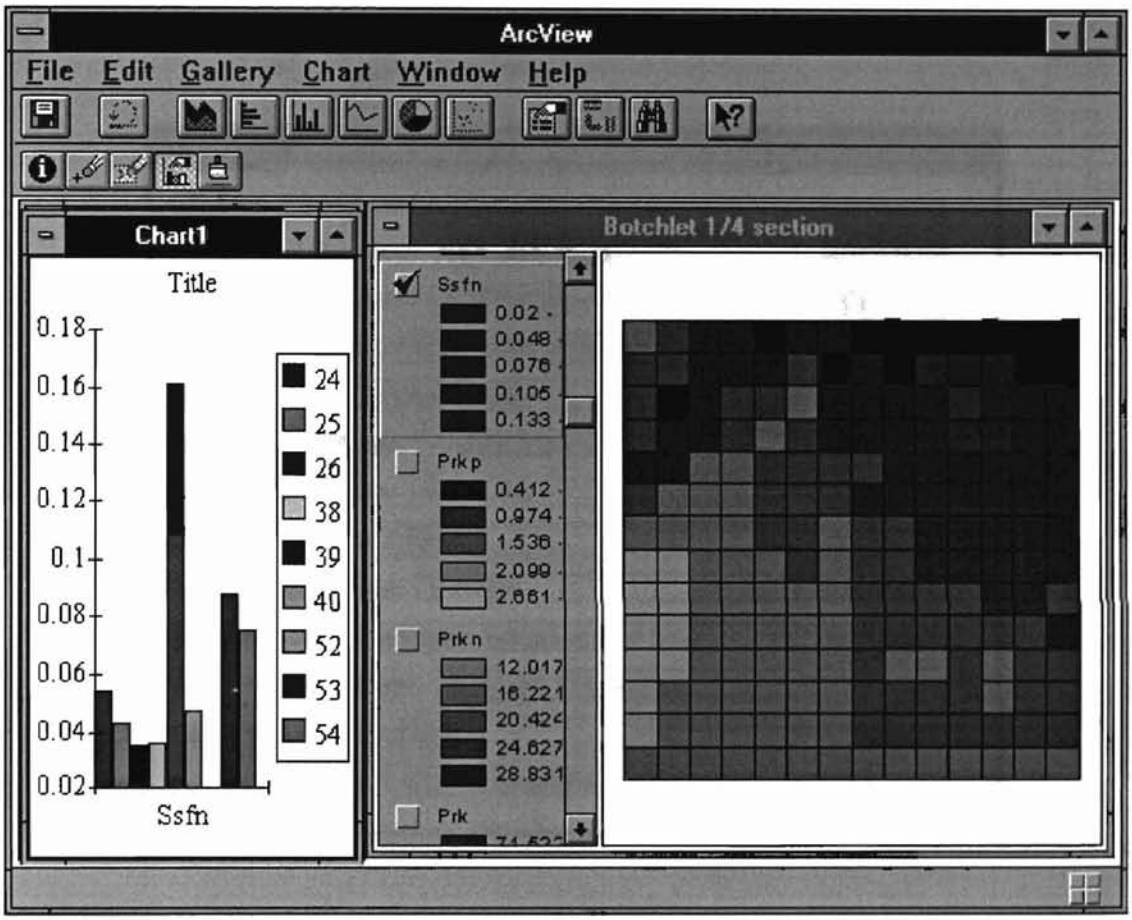
C.17 EPICView Screen Display With the Field View and Selected Cells.



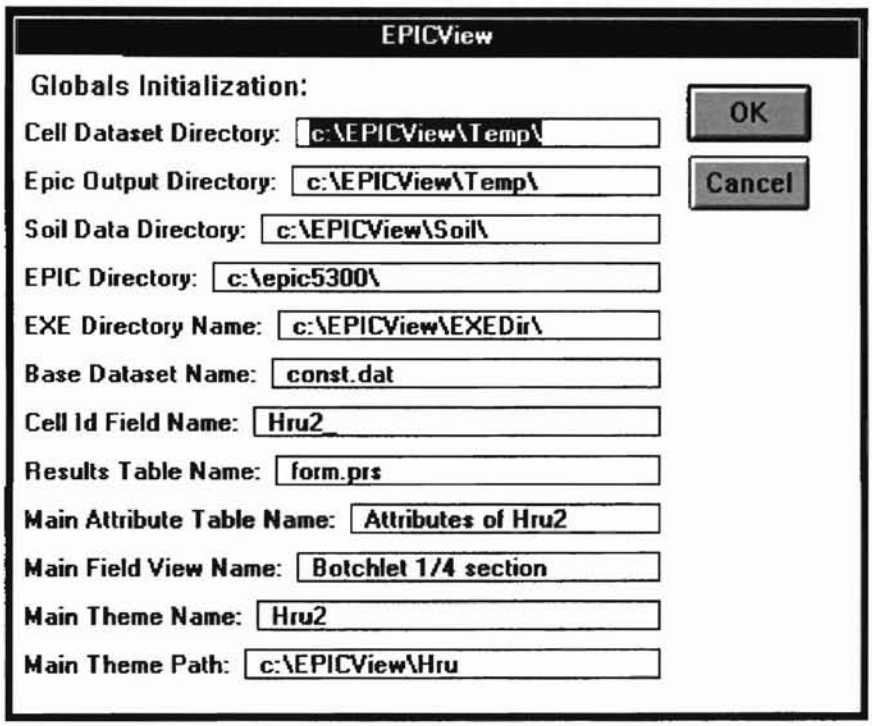
C.18 EPICView Screen Displaying the Final Results in the Form of New Themes.



C.19 EPICView Screen Displaying the Parsed Output Table.



C.20 EPICView Screen Displaying Chart Based on Output Table.



C.21 Multi Input Screen to Store Various Directory Paths.

APPENDIX D

DATA VARIABLES FROM EPIC INPUT DATASET

D.1 Constant Data Tool

No.	Variable	Description	Value
1.	ACW	Wind erosion adjustment factor 0, wind erosion shut off; 1, normal wind erosion considered	1
2.	BUS (1-4)	Four Parameter estimates for MUSI erosion equation *	0
3.	CO2*	Carbon-dioxide conc. in the atmosphere Default to 350 ppm (current level)	350
4.	CSALT*	Conc. of salt in irrigation water For future use in salinity submodel. Default to 0.	0
5.	DRV	Equation for water erosion Options [0-5]; Refer User's Guide for equations Default 2 for USLE	2
6.	IDA	Beginning day of simulation	1
7.	IET	Potential Evapotranspiration equation * Options [0-4], Refer User's guide for equations. Default 0 for Penman-Monteith equation	0
8.	IGRAF	Graphic display on/off 0, display off; 1, display on	0
9.	IHUS	Automatic heat unit scheduling 0, Normal operation; 1, automatic heat unit scheduling	0
10.	IMO	Beginning month of simulation	1
11.	ISCN	Stochastic CN Estimator code*	0
12.	ISTA	Static soil code* 0, Varying soil profile; 1, static soil	0
13.	ITYP	Peak rate estimation code* Options [0-4], Refer User's guide	0
14.	IYR	Beginning Year of simulation	1
15.	LPYR	Leap year considered* 0, consider; 1, ignore	0
16.	NBYR	Number of years of simulation duration	5
17.	PEC	Erosion control practice factor 0, Total erosion control; 1, no erosion control practices	1
18.	RTN*	Number of years of cultivation before simulation starts Values over 100 years result in little change in N values	100

* May be left blank or zero if unknown

D.2 Weather Data Tool

No.	Variable	Description	Value
1.	CF*	Climatic factor for wind erosion Refer wind erosion component (EPIC manual, Vol. I) for values	0
2.	IGN	Number of times the random number generator cycles*	0
3.	IGSD*	Day weather stops generating the same weather	0
4.	Lat Long	UTIL command - LOCWEAT < lat long>; for EPIC supplied weather data	
5.	NGN	Weather input code Options [0-5, 23, 2345] Refer User's guide 0, generates all weather ; 2345 read all weather	0
6.	RCN*	Average concentration of N in rainfall	0.8
7.	SNO*	Water content of snow at start of simulation	0
8.	SWV*	Power of modified exponential distribution of wind speed. Range (0.3 -0.5). Default to 0.5 as recommended	0.5

D.3 Spatial Data Tool

No.	Variable	Description	Value
1.	ANG*	Clockwise angle of field length from north (deg.)	0
2.	APM*	Peak runoff rate - Rainfall energy adjustment factor Value of 1 gives satisfactory results. Default to 1	1
3.	CHD	Channel Depth	0.1
4.	CHL*	Distance from outlet to most distant point on watershed	0
5.	CHN*	Channel roughness factor (Manning's N)	0
6.	CHS*	Average channel slope	0
7.	CN2	Runoff curve number (antecedent moisture condition 2(moist)) Help: Present the hydrologic groups existing in the management zones. And list the Runoff Curve Numbers for Hydrologic soil-cover complexes of Appd. E.2.A in User's Guide	
8.	ELEV	Average watershed elevation Need to pull the value from the elevation field of the attribute table of elevation coverage	
9.	FL*	Field length (Km or miles)	0
10.	FW*	Field width	0
11.	S	Slope steepness (%) Need to pull the value from the slope field of the attribute table of the coverage	
12.	SL	Slope length Run for sensitivity using $\sqrt{2}$ resolution, 2 resolution, resolution values	
13.	SN*	Surface roughness factor (Manning's N). Refer User's guide for suggested values.	0

14.	SOIL^	Prompt for <EPIC> dataset and use UTIL command GETSOIL # (# refers to soil ID from Appd. E.4). Or prompt for <User> dataset. Pull the soil series name from the attribute table and lookup the file containing the data (Soils 5 database).	
15.	STD*	Standing dead crop residue (T/ha or t/ac)	
16.	WSA	Watershed drainage area (ha) WSA = resolution ^{2} / 10000	
17.	YLT	Latitude of watershed (degrees); for daylength estimation	

* May be left blank or zero if unknown
^ Not an EPIC variable

D.4 Output Options Tool

No.	Variable	Description	Value
1.	ICODE	Output Conversion Code 0, Metric (default); 1, Metric; 2, English	2
2.	IPD	The print code to select type of output Range [1-9], Refer EPIC User's Guide	5
3.	NIPD	Printout interval 0, allows output every day or year with management operations. 1, allows output every day or year, but will not allow management operations to be printed as they occur.	0

* May be left blank or zero if unknown

D.5 Management Tool

(i) Management Options Tool

No.	Variable	Description	Value
1.	ARMN*	Minimum single application for automatic irrigation If unknown or rigid irrigation selected (NIRR= 1), set ARMN =0	0
2.	ARMX*	Maximum single application volume for automatic irrigation (mm) This is the amount applied if rigid automatic irrigation is selected. 0 if unknown	20
3.	BFT	N stress factor to trigger automatic fertilizer	0
4.	BIR	Water stress factor to trigger irrigation automatically. 0.85 means that irrigation will be triggered when the biomass production on that day is less than 85% of the potential biomass that could have been produced had water been available	0.85
5.	DRT	Time required for drainage system to reduce plant stress 0, if drainage not considered	
6.	EFI	Irrigation runoff ratio	0
7.	FMX	Maximum annual N fertilizer application 0, defaults to 200kg/ha	0
8.	FNP	Amount of fertilizer (IDFT) per automatically scheduled application 0, for manual fertilizer option	0
9.	IDFT	Fertilizer ID # for fertigation or automatic fertilization 0, no automatic fertilization or fertigation	0
10.	IDR*	Drainage code	0
11.	IFA*	Manual fertilizer application interval	0
12.	IFD*	Furrow dike code	0
13.	IFFR*	Automatic fertilizer rigidity code	0

14.	IRI*	Minimum application interval for automatic irrigation. Center pivot system - 3 days	3
15.	IRR*	Irrigation code Options 0-3. Botchlet has center pivot sprinkler, 1	1
16.	LM*	Liming code	0
17.	NIRR	Rigidity of irrigation code 0, flexible application; 1. rigid application	1
18.	NRO	Crop rotation duration Range (1-30 years)	1
19.	VIMX*	Maximum irrigation volume for each crop 24 inches (600mm) assumed to be applied for each crop per year	600

* May be left blank or zero if unknown

(ii) Management Operations Data Tool

No.	Variable	Description	Value
1.	COD	Management operation code Specify type of tillage, field equipment characteristics stored in EPIC's tillage file (Classtill.dat). Refer User's guide (Section D.3)	
2.	CRP	Crop ID # Specify type of crop from EPIC's crop file (Clascrop.dat)	
3.	DAY	Day of the operation	
4.	FAP	Fertilizer application rate (kg/ha)	
5.	FN	Fertilizer ID #	
6.	GRZ	Grazing duration in days	
7.	HUSC	Heat unit scheduling. Time of this operation as a fraction of the growing season or of the year. If no crop is growing; fraction of annual heat units accumulated using 0^{o} as the base temperature	
8.	IA	Irrigation volume (mm) Specify for manual irrigation	
9.	MON	Month of the operation	
10.	PAR	Pesticide application rate(kg/ha of active ingredient)	
11.	PCF	Pest control factor Default to 1	1
12.	PHU	Potential heat units	
13.	PST	Pesticide ID # Specify pesticide type from EPIC's file (pest5300.dat)	
14.	WSF	Plant stress factor 0.85, usually turned on for automatic irrigation at planting time	0.85

APPENDIX E

FORMAT OF VARIOUS OUTPUT FILES

E.1 A Sample EPIC Input Dataset (Form#.dat).

```
 ! Input dataset with different variable values.
! Title.
CADDO                                    21:06  4jun96
Field3
 ! Different variable values.
Wea:  22 OK WEATHERFORD  wi:  22 OK WEATHERFORD
  3  91   1   1 05  0  0  0  0  0  0  0  1  0  0  0
   .34   67.0                          1.0  35.12  236.2     .0
    .8  100.0  350.0          .0   .100
  82.0  .0160   1.00     2.
   .00    .00   8.00
 ! Weather data begins.
 8.97 12.12 17.08 23.13 27.43 32.42 35.26 34.67 30.12 24.27 16.01 10.73
-3.99 -1.34  2.62  8.83 13.73 18.75 21.13 20.28 16.05  9.86  2.87 -1.97
 7.53  7.34  7.14  5.66  4.73  4.09  3.58  3.63  5.02  5.48  6.23  6.68
 5.82  5.46  5.54  5.12  4.16  3.32  2.44  2.70  4.23  4.86  5.29  5.21
 18.9  24.9  40.2  51.3 126.2 102.8  64.2  71.6  72.0  70.7  33.5  19.2
  7.1   7.6   9.4  10.7  21.3  20.3  14.5  15.2  19.3  22.1  10.2   8.4
  .89   .81   .66   .97  2.81  3.24  1.39  1.57  2.60  2.01  1.25  1.77
 .080  .100  .130  .150  .220  .200  .150  .180  .140  .110  .100  .090
 .280  .360  .290  .370  .410  .370  .340  .250  .310  .350  .350  .240
 3.10  3.92  4.80  5.77  8.42  7.23  5.74  6.00  5.06  4.49  4.00  3.28
  7.6   7.1  19.6  28.2  36.3  49.3  41.4  30.2  27.9  18.8  21.3  13.0
 257.  327.  420.  514.  558.  636.  629.  586.  498.  377.  296.  244.
  .63   .63   .50   .53   .64   .60   .55   .54   .56   .56   .56   .61
   .00    .00    .00    .00
   .50    .00   1.00
 5.72  6.00  6.76  6.60  6.35  6.09  5.14  5.01  5.23  5.32  5.42  5.62
 16.0  14.0  11.0  10.0   7.0   4.0   3.0   3.0   6.0   9.0  12.0  14.0
  8.0   9.0   7.0   7.0   6.0   3.0   3.0   4.0   6.0   7.0   8.0   8.0
  5.0   6.0   5.0   5.0   4.0   3.0   4.0   5.0   6.0   5.0   4.0   5.0
  3.0   3.0   4.0   5.0   4.0   3.0   4.0   4.0   5.0   3.0   3.0   3.0
  3.0   4.0   4.0   5.0   6.0   5.0   7.0   6.0   5.0   3.0   3.0   2.0
  1.0   3.0   3.0   3.0   4.0   5.0   5.0   4.0   3.0   2.0   2.0   1.0
  2.0   4.0   4.0   5.0   7.0   9.0   9.0   8.0   7.0   3.0   3.0   2.0
  4.0   7.0   8.0   9.0  13.0  18.0  14.0  14.0  13.0  10.0   7.0   6.0
 18.0  16.0  19.0  20.0  27.0  31.0  27.0  28.0  28.0  27.0  20.0  18.0
 12.0   9.0   9.0   9.0   8.0  10.0  13.0  12.0   9.0  13.0  13.0  12.0
  7.0   5.0   4.0   4.0   3.0   3.0   5.0   5.0   4.0   5.0   7.0   6.0
  4.0   3.0   3.0   2.0   2.0   1.0   2.0   2.0   2.0   2.0   3.0   3.0
  3.0   3.0   3.0   3.0   2.0   1.0   1.0   1.0   1.0   2.0   3.0   4.0
  2.0   3.0   3.0   2.0   1.0   1.0   1.0   1.0   1.0   2.0   3.0   4.0
  4.0   5.0   5.0   4.0   3.0   1.0   1.0   1.0   2.0   2.0   4.0   6.0
  8.0   8.0   8.0   5.0   3.0   2.0   1.0   1.0   2.0   5.0   6.0   7.0
   .20
 ! Soil data begins.
  .010   .660  1.067  1.372  1.778
 1.655  1.655  1.609  1.623  1.641
  .054   .054   .164   .130   .067
  .139   .139   .260   .221   .164
```

```
 81.8   81.8   58.7   65.8   70.9
 12.4   12.4   17.3   15.7   20.6

  5.8    5.8    5.8    5.8    6.2

  .44    .44    .15    .06    .06

  5.7    5.7   15.2   11.4    5.2
   .5     .5    3.1    3.1    3.1



 1.68   1.68   1.74   1.72   1.67

50.00  50.00   1.17   3.14  20.42


! Crop rotation parameters.
 1  11   3   0   0   0   0   0
  .85    .00  600.00    .00  20.00    .00    .00     .0    .00    .00
! Management practices begin.
 3  25  33
 4   8  71  52     502.57 1290.32
 4  10   2   2    1500.00
 5  12  71  63     125.64 1290.32
 9  15  51                                    1.00
 9  15  41
 9  16  28
! Comments are not allowed in the input dataset. They are put here to facilitate
! readability.
```

E.2 A Sample Management Practices Batch File (Mgmt#.utl).

```
! Cell specific Management Practices batch file which will be loaded to cell's input dataset
! at the time of running model on the cell's dataset.
!  7   1
! Above line stores total operations entered and total crop rotations.
! Set of management operations.
MON(1)  3
DAY(1)  25
COD(1)  33
MON(2)  4
DAY(2)  8
COD(2)  71
FN(2)  52
FAP(2)  448.36E
FDP(2)  50.80E
HUSC(2)  0
MON(3)  4
DAY(3)  10
COD(3)  2
CRP(3)  2
GRZ(3)  2
PHU(3)  1500
MON(4)  5
DAY(4)  12
COD(4)  71
FN(4)  63
FAP(4)  112.09E
FDP(4)  50.80E
HUSC(4)  0
MON(5)  9
DAY(5)  15
COD(5)  51
HUSC(5)  1.0
MON(6)  9
DAY(6)  15
COD(6)  41
MON(7)  9
DAY(7)  16
COD(7)  28
 ! Default management related variables.
NRO   1
NIRR  1
IRR   1
IRI   3
IFA   0
LM    0
IFD   0
IDR   0
IFFR  0
IDFT  0
BIR   .85
```

```
EFI  0
VIMX  600
ARMN  0
ARMX  20
BFT  0
FNP  0
FMX  0
DRT  0
FDSF  0
! Comments are allowed in this file.
! Comments start with " !".
```

E.3 A Sample Spatial Data Batch File (Form#.utl).

```
! This File contains the spatial attributes of each cell. This is loaded into the cell's input
! dataset at the time of running the model on the cell.
! Following line provides the path for the user specified soil files.
@c:\EPICView\Soil\DoB.utl
ELEV    236.2
S       0.016
WSA     0.3364
CN2     67
SL      82.0244
! Following line provides the path for the cell specific management practices batch file.
@c:\EPICView\Temp\mgmt25.utl
! Comments are allowed in this file.
```

E.4 A Sample Constant Data Batch File (Const.utl).

```
! Written by Weather Tool.
LOCWEAT  35.12 98.35
YLT      35.12
! Written by Constant Data Tool.
TITLE(1)  CADDO COUNTY, OK, SHERRY/BOTCHLET
TITLE(2)  Field
NBYR     3
IYR      91
IMO      1
IDA      1
NIPD     0
IPD      5
NGN      0
IGN      0
IGSD     0
LPYR     0
IET      0
ISCN     0
IGRAF    0
ICODE    1
ITYP     0
ISTA     0
IHUS     0
CHL      0
CHS      0
CHN      0
SN       0
APM      1
SNO      0
RCN      0.8
RTN      100
CO2      350
CSALT    0
CHD      0.1
PEC      1
DRV      2
BUS(1)   0
BUS(2)   0
BUS(3)   0
BUS(4)   0
FL       0
FW       0
ANG      0
STD      0
ACW      1
```

E.5 A Sample Output Options Batch File (Prnt.utl).

```
! Initialize KD(1) and KM(1) with 0 so that EPIC dumps default variables in ".epd" and
! ".epm" files respectively, if a user chooses to get these output files from Output Options
! Tool.
KD(1)   0
KM(1)   0
! Yearly variables are set as per those selected by a user. The remaining are set to 0.
! A user can choose atmost 30 variables at one time.
KY(1)   46
KY(2)   47
KY(3)   12
KY(4)   11
KY(5)   56
KY(6)   41
KY(7)   50
KY(8)   31
KY(9)   16
KY(10)   40
KY(11)   51
KY(12)   39
KY(13)   23
KY(14)   38
KY(15)  0
KY(16)  0
KY(17)  0
KY(18)  0
KY(19)  0
KY(20)  0
KY(21)  0
KY(22)  0
KY(23)  0
KY(24)  0
KY(25)  0
KY(26)  0
KY(27)  0
KY(28)  0
KY(29)  0
KY(30)  0
```

E.6 A Sample Batch File for Completion of Datasets (create.bat).

```
c:\epic5300\util epic c:\EPICView\Temp\form24.dat @c:\EPICView\Temp\form24.utl
c:\epic5300\util epic c:\EPICView\Temp\form25.dat @c:\EPICView\Temp\form25.utl
c:\epic5300\util epic c:\EPICView\Temp\form26.dat @c:\EPICView\Temp\form26.utl
c:\epic5300\util epic c:\EPICView\Temp\form38.dat @c:\EPICView\Temp\form38.utl
c:\epic5300\util epic c:\EPICView\Temp\form39.dat @c:\EPICView\Temp\form39.utl
c:\epic5300\util epic c:\EPICView\Temp\form40.dat @c:\EPICView\Temp\form40.utl
c:\epic5300\util epic c:\EPICView\Temp\form52.dat @c:\EPICView\Temp\form52.utl
c:\epic5300\util epic c:\EPICView\Temp\form53.dat @c:\EPICView\Temp\form53.utl
c:\epic5300\util epic c:\EPICView\Temp\form54.dat @c:\EPICView\Temp\form54.utl
c:\epic5300\util prnt prnt5300.dat @c:\EPICView\Temp\prnt.utl
c:\EPICView\Temp\runepic.bat
```

E.7 A Sample Batch File for Running EPIC and Parsing output (runepic.bat).

```
c:\epic5300\ewq c:\EPICView\Temp\form24 c:\EPICView\Temp\
c:\epic5300\ewq c:\EPICView\Temp\form25 c:\EPICView\Temp\
c:\epic5300\ewq c:\EPICView\Temp\form26 c:\EPICView\Temp\
c:\epic5300\ewq c:\EPICView\Temp\form38 c:\EPICView\Temp\
c:\epic5300\ewq c:\EPICView\Temp\form39 c:\EPICView\Temp\
c:\epic5300\ewq c:\EPICView\Temp\form40 c:\EPICView\Temp\
c:\epic5300\ewq c:\EPICView\Temp\form52 c:\EPICView\Temp\
c:\epic5300\ewq c:\EPICView\Temp\form53 c:\EPICView\Temp\
c:\epic5300\ewq c:\EPICView\Temp\form54 c:\EPICView\Temp\
c:\EPICView\EXEDir\parse.exe
```

APPENDIX F

CODE FOR EACH USER INTERFACE SCREEN

```
'Weather.frm
' This form allows a user to enter the latitudes and longitudes of the field or specify the
' path of the weather file. This information is stored in "const.utl" file.

Option Explicit

Dim cellDatasetDir As String
Dim epicDir As String
Dim soilDir As String
Dim epicOutputDir As String
Dim exeDir As String
Dim fileNum As Integer
Dim fileName As String
Dim Latitude As Single
Dim Longitude As Single

Dim validateText As String 'To use as a buffer for validation.

Private Sub cmdCancel_Click()
  Dim resp As Integer

  resp = MsgBox("Do you wish to close?", vbYesNo + vbCritical + vbDefaultButton2, "EPIC-View")
  If resp = vbYes Then
    End
  End If
End Sub

Private Sub cmdOk_Click()
  Dim resp As Integer

  'Store values if a user chooses to.
  resp = MsgBox("Do you wish to store?", vbYesNo + vbCritical + vbDefaultButton2, "EPIC-View")
  If resp = vbYes Then
    If optEPICFile(0).Value = True Then
      If ((txtLat.Text <> "") And (txtLong.Text <> "")) Then
        fileNum = FreeFile
        Open cellDatasetDir + "const.utl" For Output As fileNum
        Print #fileNum, "LOCWEAT   " + txtLat.Text + " " + txtLong.Text
        Print #fileNum, "YLT       " + txtLat.Text
        Close fileNum
        End
      End If
    ElseIf optMyFile(1).Value = True Then
      If (txtPath.Text <> "") Then
        fileNum = FreeFile
        Open cellDatasetDir + "const.utl" For Output As fileNum
        Print #fileNum, "@" + txtPath.Text
        Close fileNum
        End
      End If
```

```
      End If
    End If
End Sub


Private Sub Form_Load()
    'Get directory paths.
    fileNum = FreeFile
    Open "c:\EVPaths.txt" For Input As fileNum
    Input #fileNum, cellDatasetDir, epicOutputDir, soilDir, epicDir, exeDir
    Close fileNum
End Sub

Private Sub optEPICFile_Click(Index As Integer)
    txtLat.Visible = True
    txtLong.Visible = True
    txtPath.Visible = False
    lblLat.Visible = True
    lblLong.Visible = True
    lblPath.Visible = False
End Sub


Private Sub optMyFile_Click(Index As Integer)
    txtLat.Visible = False
    txtLong.Visible = False
    txtPath.Visible = True
    lblLat.Visible = False
    lblLong.Visible = False
    lblPath.Visible = True
End Sub



' ValidateData
' Author: Anoop Govil
' Date: June 12, 1996
'-----------------------------------------------
' validateData Subroutine:
'  - Validates the value entered by a user so
'  - that it lies in the range of real numbers.
'-----------------------------------------------
Public Sub validateData()
    Dim length As Integer
    Dim resp As Integer
    Dim start As Integer
    Dim alreadyDecimal As Integer
    Dim chr As String
    chr = 0
    length = 1
    alreadyDecimal = 0


    Do
```

```
    chr = Mid(validateText, length, 1)
    ' If character other than a number or a decimal.
    If (chr <> "." And chr <> "0" And chr <> "1" And chr <> "2" And chr <> "3" And chr <> "4" _
      And chr <> "5" And chr <> "6" And chr <> "7" And chr <> "8" And chr <> "9") Then
      Beep
      If (length = 1) Then
        start = 2
      Else
        start = 1
      End If
      If (Len(validateText) > 0) Then
        validateText = Mid(validateText, start, Len(validateText) - 1)
      End If
      Exit Do
    ' If character is a decimal.
    ElseIf (chr = "." And alreadyDecimal = 0) Then
      alreadyDecimal = 1
    ' If character is a second decimal point.
    ElseIf (chr = "." And alreadyDecimal = 1) Then
      Beep
      If (length = 1) Then
        start = 2
      Else
        start = 1
      End If
      If (Len(validateText) > 0) Then
        validateText = Mid(validateText, start, Len(validateText) - 1)
      End If
      Exit Do
    End If
    length = length + 1
  Loop While (length <= Len(validateText))
End Sub

Private Sub txtLat_Change()
  validateText = txtLat.Text
  validateData
  txtLat.Text = validateText
End Sub


Private Sub txtLong_Change()
  validateText = txtLong.Text
  validateData
  txtLong.Text = validateText
End Sub
```

```
'Soil.frm
' This form allows a user to select a soil which becomes generic for the whole field and
' also select the run off curve number. This information is stored in the "const.utl" file.

Option Explicit

Dim soilNames(1000) As String
Dim soilCodes(1000) As Integer
Dim landUse(20) As String
Dim coverTrtmnt(5) As String
Dim hydCondition(5) As String
Dim hydSoilGrp(5) As String
Dim curveNumber(0 To 11, 0 To 3, 0 To 3, 0 To 4) As Integer
Dim fileNum As Integer
Dim cellDatasetDir As String
Dim epicOutputDir As String
Dim soilDir As String
Dim epicDir As String
Dim ctr1 As Integer
Dim ctr2 As Integer
Dim ctr3 As Integer
Dim ctr4 As Integer

Private Sub cboCoverTrtmnt_Change()
   Dim Counter As Integer
   Counter = 0
   'Get the index for selected cover treatment.
   Do
      If (coverTrtmnt(Counter) = cboCoverTrtmnt.Text) Then
         Exit Do
      End If
      Counter = Counter + 1
   Loop While (Counter < 5)
   If Counter < 5 Then
      ctr2 = Counter
      'Set curve number.
      txtCurveNumber.Text = curveNumber(ctr1, ctr2, ctr3, ctr4)
   End If
End Sub

Private Sub cboCoverTrtmnt_Click()
   'Get the index for selected cover treatment.
   ctr2 = cboCoverTrtmnt.ListIndex
   'Set curve number.
   txtCurveNumber.Text = curveNumber(ctr1, ctr2, ctr3, ctr4)
End Sub


Private Sub cboHydCondition_Change()
   Dim Counter As Integer
   Counter = 0
   'Get the index for selected hydrologic condition.
   Do
      If (hydCondition(Counter) = cboHydCondition.Text) Then
```

```
        Exit Do
      End If
      Counter = Counter + 1
    Loop While (Counter < 5)
    If Counter < 5 Then
      ctr3 = Counter
      'Set the curve number.
      txtCurveNumber.Text = curveNumber(ctr1, ctr2, ctr3, ctr4)
    End If
End Sub

Private Sub cboHydCondition_Click()
    'Get the index for selected hydrologic condition.
    ctr3 = cboHydCondition.ListIndex
    'Set the curve number.
    txtCurveNumber.Text = curveNumber(ctr1, ctr2, ctr3, ctr4)
End Sub

Private Sub cboHydSoilGrp_Change()
    Dim Counter As Integer
    Counter = 0
    'Get the index for selected hydrologic soil group.
    Do
    If (hydSoilGrp(Counter) = cboHydSoilGrp.Text) Then
      Exit Do
    End If
    Counter = Counter + 1
    Loop While (Counter < 5)
    If Counter < 5 Then
      ctr4 = Counter
      'Set the curve number.
      txtCurveNumber.Text = curveNumber(ctr1, ctr2, ctr3, ctr4)
    End If
End Sub

Private Sub cboHydSoilGrp_Click()
    'Get the index for selected hydrologic soil group.
    ctr4 = cboHydSoilGrp.ListIndex
    'Set the curve number.
    txtCurveNumber.Text = curveNumber(ctr1, ctr2, ctr3, ctr4)
End Sub

Private Sub cboLandUse_Change()
    Dim Counter As Integer
    Counter = 0
    'Get the index for selected land use.
    Do
      If (landUse(Counter) = cboLandUse.Text) Then
        Exit Do
      End If
      Counter = Counter + 1
    Loop While (Counter < 5)
    If Counter < 5 Then
      ctr1 = Counter
```

```
      'Set the curve number.
      txtCurveNumber.Text = curveNumber(ctr1, ctr2, ctr3, ctr4)
   End If
End Sub

Private Sub cboLandUse_Click()
   'Get the index for selected land use.
   ctr1 = cboLandUse.ListIndex
   'Set the curve number.
   txtCurveNumber.Text = curveNumber(ctr1, ctr2, ctr3, ctr4)
End Sub


Private Sub cmdCancel_Click()
   Dim resp As Integer

   resp = MsgBox("Do you wish to close?", vbYesNo + vbCritical + vbDefaultButton2, "EPIC-View")
   If resp = vbYes Then
      End
   End If
End Sub


Private Sub cmdOK_Click()
   Dim Counter As Integer
   Dim resp As Integer

   'Get the soil code.
   Counter = 0
   Do
      If (UCase(cboSoilNames.Text) = soilNames(Counter)) Then
         Exit Do
      End If
      Counter = Counter + 1
   Loop While Counter < 1001

   'Store the soil code and curve number if a user chooses to.
   If (Counter < 1001) Then
      resp = MsgBox("Do you wish to store?", vbYesNo + vbCritical + vbDefaultButton2, "EPIC-View")
      If resp = vbYes Then
         fileNum = FreeFile
         Open cellDatasetDir + "const.utl" For Append As fileNum
         Print #fileNum, "GETSOIL   " + Str(soilCodes(Counter))
         Print #fileNum, "CN2       " + txtCurveNumber.Text
         Close fileNum
         End
      End If
   End If
End Sub

Private Sub Form_Load()
   Dim Counter As Integer
   Dim fileName As String
   Dim soilCode As Integer
```

```
Dim soilName As String
Dim c1 As Integer
Dim c2 As Integer
Dim c3 As Integer
Dim c4 As Integer
Dim ct1 As Integer
Dim ct2 As Integer
Dim ct3 As Integer
Dim ct4 As Integer
Dim value As Integer

'Get directory paths.
fileNum = FreeFile
Open "c:\EVPaths.txt" For Input As fileNum
Input #fileNum, cellDatasetDir, epicOutputDir, soilDir, epicDir
Close fileNum

'Open the soil file and create a soil names list.
fileNum = FreeFile
fileName = epicDir + "soil.lis"
Open fileName For Input As fileNum
Counter = 0
Do
   Input #fileNum, soilCodes(Counter), soilNames(Counter)
   cboSoilNames.AddItem _
         soilNames(Counter)
   Counter = Counter + 1
Loop While Not (EOF(fileNum) Or (Counter > 1000))
cboSoilNames.ListIndex = 641
Close fileNum

'Open the curve number file and create various supporting lists.
fileNum = FreeFile
fileName = epicDir + "curvenum.dat"
Open fileName For Input As fileNum
ctr1 = 0
'Create land use list.
Do
   Input #fileNum, landUse(ctr1)
   If (landUse(ctr1) <> "") Then
   cboLandUse.AddItem _
         landUse(ctr1)
   Else: Exit Do
   End If
   ctr1 = ctr1 + 1
Loop While Not (EOF(fileNum) And (ctr1 > 20))
ct1 = ctr1
cboLandUse.ListIndex = 0

'Create cover treatment list.
ctr2 = 0
Do
   Input #fileNum, coverTrtmnt(ctr2)
   If (coverTrtmnt(ctr2) <> "") Then
```

```
    cboCoverTrtmnt.AddItem _
            coverTrtmnt(ctr2)
    Else: Exit Do
    End If
    ctr2 = ctr2 + 1
Loop While Not (EOF(fileNum) And (ctr2 > 5))
ct2 = ctr2
cboCoverTrtmnt.ListIndex = 0

'Create hydrologic condition list.
ctr3 = 0
Do
    Input #fileNum, hydCondition(ctr3)
    If (hydCondition(ctr3) <> "") Then
    cboHydCondition.AddItem _
            hydCondition(ctr3)
    Else: Exit Do
    End If
    ctr3 = ctr3 + 1
Loop While Not (EOF(fileNum) And (ctr3 > 5))
ct3 = ctr3
cboHydCondition.ListIndex = 0

'Create hydrologic soil group list.
ctr4 = 0
Do
    Input #fileNum, hydSoilGrp(ctr4)
    If (hydSoilGrp(ctr4) <> "") Then
    cboHydSoilGrp.AddItem _
            hydSoilGrp(ctr4)
    Else: Exit Do
    End If
    ctr4 = ctr4 + 1
Loop While Not (EOF(fileNum) And (ctr4 > 5))
ct4 = ctr4
cboHydSoilGrp.ListIndex = 0
c1 = 0
c2 = 0
c3 = 0
c4 = 0

'Load the values in the curve number array.
Do
    If (c1 = ct1) Then
        Exit Do
    End If
    c2 = 0
    Do
        If (c2 = ct2) Then
            Exit Do
        End If
        c3 = 0
        Do
            If (c3 = ct3) Then
```

```
            Exit Do
          End If
          c4 = 0
          Do
            If (c4 = ct4) Then
            Exit Do
            End If
            Input #fileNum, value
            curveNumber(c1, c2, c3, c4) = value
            c4 = c4 + 1
          Loop While Not (EOF(fileNum) And (c4 < ct4))
          c3 = c3 + 1
        Loop While Not (EOF(fileNum) And (c3 < ct3))
        c2 = c2 + 1
      Loop While Not (EOF(fileNum) And (c2 < ct2))
      c1 = c1 + 1
    Loop While Not (EOF(fileNum) And (c1 < ct1))
    Close fileNum
    txtCurveNumber.Text = curveNumber(ctr1, ctr2, ctr3, ctr4)
End Sub
```

```
'constdat.bas
' This form allows a user to modify the values of variables which remain constant for the
' whole field. They are stored in the file "const.utl" and later a constant EPIC input
' dataset "const.dat" is created and replicated for all the cells present in the gridded
' coverage.

Option Explicit

' directory paths.
Global cellDatasetDir As String
Global epicOutputDir As String
Global soilDir As String
Global epicDir As String

Global totalCells As Integer
Global validateText As String 'To use as a buffer for validation.

Declare Function GetModuleUsage% Lib "Kernel" (ByVal hModule%)

'-----------------------------------------------
' WaitShell
' Author: Anoop Govil
' Date: May 21, 1996
'-----------------------------------------------
' WaitShell Subroutine:
'  - Makes a synchronous call.
'-----------------------------------------------
Public Sub WaitShell(ByVal AppName As String)
  Dim hMod As Integer
  hMod = Shell(AppName, 1)
  If (Abs(hMod) > 32) Then
    While (GetModuleUsage(hMod))
      DoEvents
    Wend
  Else
    Debug.Print "Unable to start " & AppName
  End If

End Sub

'-----------------------------------------------
' ValidateData
' Author: Anoop Govil
' Date: June 12, 1996
'-----------------------------------------------
' validateData Subroutine:
' - Validates the value entered by a user so
' - that it lies in the range of real numbers.
'-----------------------------------------------
Public Sub validateData()
   Dim length As Integer
   Dim resp As Integer
   Dim start As Integer
   Dim alreadyDecimal As Integer
```

```
  Dim chr As String
  chr = 0
  length = 1
  alreadyDecimal = 0


  Do
     chr = Mid(validateText, length, 1)
     ' If character other than a number or a decimal.
     If (chr <> "." And chr <> "0" And chr <> "1" And chr <> "2" And chr <> "3" And chr <> "4" _
        And chr <> "5" And chr <> "6" And chr <> "7" And chr <> "8" And chr <> "9") Then
        Beep
        If (length = 1) Then
           start = 2
        Else
           start = 1
        End If
        If (Len(validateText) > 0) Then
           validateText = Mid(validateText, start, Len(validateText) - 1)
        End If
        Exit Do
     ' If character is a decimal.
     ElseIf (chr = "." And alreadyDecimal = 0) Then
        alreadyDecimal = 1
     ' If character is a second decimal point.
     ElseIf (chr = "." And alreadyDecimal = 1) Then
        Beep
        If (length = 1) Then
           start = 2
        Else
           start = 1
        End If
        If (Len(validateText) > 0) Then
           validateText = Mid(validateText, start, Len(validateText) - 1)
        End If
        Exit Do
     End If
     length = length + 1
  Loop While (length <= Len(validateText))
End Sub
```

```
'Constdat.frm
' This form allows a user to modify the values of variables which remain constant for the
' whole field. They are stored in the file "const.utl" and later a constant EPIC input
' dataset "const.dat" is created and replicated for all the cells present in the gridded
' coverage.

Option Explicit

Dim fileNum As Integer

Private Sub cmdCancel_Click()
   Dim resp As Integer
   resp = MsgBox("Do you wish to close?", vbYesNo + vbCritical + vbDefaultButton2, "EPIC-View")
   If resp = vbYes Then
      End
   End If
End Sub

Private Sub cmdHelp_Click()
   frmConstDat1Hlp.Show
End Sub

Private Sub cmdMore_Click()
   frmConstDatMore.Show
End Sub

Private Sub cmdOk_Click()
   Dim resp As Integer
   Dim runCommand As String
   Dim fileName As String
   Dim Counter As Integer

   ' Write data to file if a user chooses to.
   resp = MsgBox("Do you wish to load constant variables in the datasets?", vbYesNo + vbCritical +
vbDefaultButton2, "EPIC-View")
   If resp = vbYes Then
      Hide
      fileNum = FreeFile
      Open cellDatasetDir + "const.utl" For Append As fileNum

      Print #fileNum, "TITLE(1)  " + txtTitle1.Text
      Print #fileNum, "TITLE(2)  Field"
      Print #fileNum, "NBYR     " + txtNBYR.Text
      Print #fileNum, "IYR      " + txtIYR.Text
      Print #fileNum, "IMO      " + txtIMO.Text
      Print #fileNum, "IDA      " + txtIDA.Text
      Print #fileNum, "NIPD     " + txtNIPD.Text
      Print #fileNum, "IPD      " + txtIPD.Text
      Print #fileNum, "NGN      " + txtNGN.Text
      Print #fileNum, "IGN      " + frmConstDatMore.txtIGN.Text
      Print #fileNum, "IGSD     " + frmConstDatMore.txtIGSD.Text
      Print #fileNum, "LPYR     " + txtLPYR.Text
      Print #fileNum, "IET      " + txtIET.Text
      Print #fileNum, "ISCN     " + frmConstDatMore.txtISCN.Text
```

```
    Print #fileNum, "IGRAF    " + frmConstDatMore.txtIGRAF.Text
    Print #fileNum, "ICODE    " + txtICODE.Text
    Print #fileNum, "ITYP     " + frmConstDatMore.txtITYP.Text
    Print #fileNum, "ISTA     " + frmConstDatMore.txtISTA.Text
    Print #fileNum, "IHUS     " + txtIHUS.Text
    Print #fileNum, "CHL      " + frmConstDatMore.txtCHL.Text
    Print #fileNum, "CHS      " + frmConstDatMore.txtCHS.Text
    Print #fileNum, "CHN      " + frmConstDatMore.txtCHN.Text
    Print #fileNum, "SN       " + frmConstDatMore.txtSN.Text
    Print #fileNum, "APM      " + frmConstDatMore.txtAPM.Text
    Print #fileNum, "SNO      " + frmConstDatMore.txtSNO.Text
    Print #fileNum, "RCN      " + frmConstDatMore.txtRCN.Text
    Print #fileNum, "RTN      " + frmConstDatMore.txtRTN.Text
    Print #fileNum, "CO2      " + frmConstDatMore.txtCO2.Text
    Print #fileNum, "CSALT    " + frmConstDatMore.txtCSALT.Text
    Print #fileNum, "CHD      " + frmConstDatMore.txtCHD.Text
    Print #fileNum, "PEC      " + txtPEC.Text
    Print #fileNum, "DRV      " + txtDRV.Text
    Print #fileNum, "BUS(1)   " + frmConstDatMore.txtBUS1.Text
    Print #fileNum, "BUS(2)   " + frmConstDatMore.txtBUS2.Text
    Print #fileNum, "BUS(3)   " + frmConstDatMore.txtBUS3.Text
    Print #fileNum, "BUS(4)   " + frmConstDatMore.txtBUS4.Text
    Print #fileNum, "FL       " + frmConstDatMore.txtFL.Text
    Print #fileNum, "FW       " + frmConstDatMore.txtFW.Text
    Print #fileNum, "ANG      " + frmConstDatMore.txtANG.Text
    Print #fileNum, "STD      " + frmConstDatMore.txtSTD.Text
    Print #fileNum, "ACW      " + txtACW.Text

    Close fileNum
    fileNum = FreeFile
    fileName = cellDatasetDir + "const.dat"
    Open fileName For Output As fileNum
    Close fileNum
    ' Load the constant variables' values to the const.dat.
    runCommand = epicDir + "util epic " + cellDatasetDir + "const.dat @" + cellDatasetDir + "const.utl"
    WaitShell (runCommand)
    ' Replicate the const.dat file for all the cells' dataset files.
    resp = MsgBox("Creating datasets for all the cells.", vbCritical, "EPIC-View")
    Counter = 1
    Do
      fileName = cellDatasetDir + "form" + Trim(Str(Counter)) + ".dat"
      FileCopy cellDatasetDir + "const.dat", fileName
      Counter = Counter + 1
    Loop While Counter <= totalCells
    resp = MsgBox("Datasets have been created.", vbInformation, "EPIC-View")
    End
  End If
End Sub

Private Sub Form_Load()
  ' Get all directory paths.
  fileNum = FreeFile
  Open "c:\EVPaths.txt" For Input As fileNum
  Input #fileNum, cellDatasetDir, epicOutputDir, soilDir, epicDir
```

```
    Close fileNum
    fileNum = FreeFile
    ' Get total number of cells in the gridded coverage.
    Open cellDatasetDir + "selected.cll" For Input As fileNum
    Input #fileNum, totalCells
    Close fileNum
End Sub

Private Sub txtACW_Change()
    validateText = txtACW.Text
    validateData
    txtACW.Text = validateText
End Sub


Private Sub txtDRV_Change()
    validateText = txtDRV.Text
    validateData
    txtDRV.Text = validateText
End Sub

Private Sub txtICODE_Change()
    validateText = txtICODE.Text
    validateData
    txtICODE.Text = validateText
End Sub

Private Sub txtIDA_Change()
    validateText = txtIDA.Text
    validateData
    txtIDA.Text = validateText
End Sub

Private Sub txtIET_Change()
    validateText = txtIET.Text
    validateData
    txtIET.Text = validateText
End Sub

Private Sub txtIHUS_Change()
    validateText = txtIHUS.Text
    validateData
    txtIHUS.Text = validateText
End Sub

Private Sub txtIMO_Change()
    validateText = txtIMO.Text
    validateData
    txtIMO.Text = validateText
End Sub

Private Sub txtIPD_Change()
    validateText = txtIPD.Text
    validateData
```

```
    txtIPD.Text = validateText
End Sub

Private Sub txtIYR_Change()
    validateText = txtIYR.Text
    validateData
    txtIYR.Text = validateText
End Sub

Private Sub txtLPYR_Change()
    validateText = txtLPYR.Text
    validateData
    txtLPYR.Text = validateText
End Sub

Private Sub txtNBYR_Change()
    validateText = txtNBYR.Text
    validateData
    txtNBYR.Text = validateText
End Sub

Private Sub txtNGN_Change()
    validateText = txtNGN.Text
    validateData
    txtNGN.Text = validateText
End Sub

Private Sub txtNIPD_Change()
    validateText = txtNIPD.Text
    validateData
    txtNIPD.Text = validateText
End Sub

Private Sub txtPEC_Change()
    validateText = txtPEC.Text
    validateData
    txtPEC.Text = validateText
End Sub
```

```
'Constmore.frm
' This form allows a user to modify the values of variables which remain constant for the
' whole field. They are stored in the file "const.utl" and later a constant EPIC input
' dataset "const.dat" is created and replicated for all the cells present in the gridded
' coverage.

Private Sub cmdHelp_Click()
   frmConstDat2Hlp.Show
End Sub

Private Sub cmdOk_Click()
   Hide
End Sub

Private Sub txtANG_Change()
   validateText = txtANG.Text
   validateData
   txtANG.Text = validateText
End Sub

Private Sub txtAPM_Change()
   validateText = txtAPM.Text
   validateData
   txtAPM.Text = validateText
End Sub

Private Sub txtBUS1_Change()
   validateText = txtBUS1.Text
   validateData
   txtBUS1.Text = validateText
End Sub

Private Sub txtBUS2_Change()
   validateText = txtBUS2.Text
   validateData
   txtBUS2.Text = validateText
End Sub

Private Sub txtBUS3_Change()
   validateText = txtBUS3.Text
   validateData
   txtBUS3.Text = validateText
End Sub

Private Sub txtBUS4_Change()
   validateText = txtBUS4.Text
   validateData
   txtBUS4.Text = validateText
End Sub

Private Sub txtCF_Change()
   validateText = txtCF.Text
   validateData
   txtCF.Text = validateText
```

```
End Sub

Private Sub txtCHD_Change()
   validateText = txtCHD.Text
   validateData
   txtCHD.Text = validateText
End Sub

Private Sub txtCHL_Change()
   validateText = txtCHL.Text
   validateData
   txtCHL.Text = validateText
End Sub

Private Sub txtCHN_Change()
   validateText = txtCHN.Text
   validateData
   txtCHN.Text = validateText
End Sub

Private Sub txtCHS_Change()
   validateText = txtCHS.Text
   validateData
   txtCHS.Text = validateText
End Sub

Private Sub txtCO2_Change()
   validateText = txtCO2.Text
   validateData
   txtCO2.Text = validateText
End Sub

Private Sub txtCSALT_Change()
   validateText = txtCSALT.Text
   validateData
   txtCSALT.Text = validateText
End Sub

Private Sub txtFL_Change()
   validateText = txtFL.Text
   validateData
   txtFL.Text = validateText
End Sub

Private Sub txtFW_Change()
   validateText = txtFW.Text
   validateData
   txtFW.Text = validateText
End Sub

Private Sub txtIGN_Change()
   validateText = txtIGN.Text
   validateData
   txtIGN.Text = validateText
```

```
End Sub

Private Sub txtIGRAF_Change()
   validateText = txtIGRAF.Text
   validateData
   txtIGRAF.Text = validateText
End Sub

Private Sub txtIGSD_Change()
   validateText = txtIGSD.Text
   validateData
   txtIGSD.Text = validateText
End Sub

Private Sub txtISCN_Change()
   validateText = txtISCN.Text
   validateData
   txtISCN.Text = validateText
End Sub

Private Sub txtISTA_Change()
   validateText = txtISTA.Text
   validateData
   txtISTA.Text = validateText
End Sub

Private Sub txtITYP_Change()
   validateText = txtITYP.Text
   validateData
   txtITYP.Text = validateText
End Sub

Private Sub txtRCN_Change()
   validateText = txtRCN.Text
   validateData
   txtRCN.Text = validateText
End Sub

Private Sub txtRTN_Change()
   validateText = txtRTN.Text
   validateData
   txtRTN.Text = validateText
End Sub

Private Sub txtSN_Change()
   validateText = txtSN.Text
   validateData
   txtSN.Text = validateText
End Sub

Private Sub txtSNO_Change()
   validateText = txtSNO.Text
   validateData
   txtSNO.Text = validateText
```

```
End Sub

Private Sub txtSTD_Change()
  validateText = txtSTD.Text
  validateData
  txtSTD.Text = validateText
End Sub

Private Sub txtSWV_Change()
  validateText = txtSWV.Text
  validateData
  txtSWV.Text = validateText
End Sub
```

```
'Mgmt.bas
' This form allows a user to enter management practices for selected cells or for the whole
' field as chosen by a user. Different operations can be selected and the operations are
' stored in the cell specific "mgmt.utl" file.

Option Explicit

Global monthSel As Integer 'For selected month.
Global daySel As Integer  'For selected day.
Global unitType(0 To 2) As String 'For selected unidt type.
Global operationCode As Integer 'For selected management operation.
Global cellFiles() As Integer 'To maintain different cell files' pointers.
Global cellIndex() As Integer 'To maintain different cell indexes.
Global cellExist() As Integer 'To maintain different flags if cell file exists.
Global selectedCells(1 To 1000) As String 'To store number of selected cells.
Global nro() As Integer 'To maintain different cells' NRO values.
Global totCellsSel As Integer
Global didCropRotation As Integer
Global currentCell As Integer
Global justDidCropRotation As Integer
Global validateText As String 'To use as a buffer for validation.

' Directory paths.
Global cellDatasetDir As String
Global epicOutputDir As String
Global soilDir As String
Global epicDir As String

' Management related variables.
Global armn As Single
Global armx As Single
Global bft As Single
Global bir As Single
Global drt As Single
Global efi As Single
Global fdsf As Single
Global fmx As Single
Global fnp As Single
Global idft As Single
Global idr As Single
Global ifa As Single
Global ifd As Single
Global iffr As Single
Global iri As Single
Global irr As Single
Global lm As Single
Global nirr As Single
Global vimx As Single


' ValidateData
' Author: Anoop Govil
' Date: June 12, 1996
'-----------------------------------------
```

```
' validateData Subroutine:
' - Validates the value entered by a user so
' - that it lies in the range of real numbers.
'-----------------------------------------------
Public Sub validateData()
   Dim length As Integer
   Dim resp As Integer
   Dim start As Integer
   Dim alreadyDecimal As Integer
   Dim chr As String
   chr = 0
   length = 1
   alreadyDecimal = 0


   Do
      chr = Mid(validateText, length, 1)
      ' If character other than a number or a decimal.
      If (chr <> "." And chr <> "0" And chr <> "1" And chr <> "2" And chr <> "3" And chr <> "4" _
         And chr <> "5" And chr <> "6" And chr <> "7" And chr <> "8" And chr <> "9") Then
         Beep
         If (length = 1) Then
            start = 2
         Else
            start = 1
         End If
         If (Len(validateText) > 0) Then
            validateText = Mid(validateText, start, Len(validateText) - 1)
         End If
         Exit Do
      ' If character is a decimal.
      ElseIf (chr = "." And alreadyDecimal = 0) Then
         alreadyDecimal = 1
      ' If character is a second decimal point.
      ElseIf (chr = "." And alreadyDecimal = 1) Then
         Beep
         If (length = 1) Then
            start = 2
         Else
            start = 1
         End If
         If (Len(validateText) > 0) Then
            validateText = Mid(validateText, start, Len(validateText) - 1)
         End If
         Exit Do
      End If
      length = length + 1
   Loop While (length <= Len(validateText))
End Sub
```

```
'Mgmt.frm
' This form allows a user to enter management practices for selected cells or for the whole
' field as chosen by a user. Different operations can be selected and the operations are
' stored in the cell specific "mgmt.utl" file.

Option Explicit

Dim month(1 To 12) As String
Dim day(1 To 31) As Integer
Dim operation(1 To 100) As String
Dim operCode(1 To 100) As Integer
Dim fileNum As Integer
Dim firstTime As Integer

Private Sub cboDay_Change()
   Dim Counter As Integer
   Counter = 1
   Do
      If (day(Counter) = cboDay.Text) Then
         Exit Do
      End If
      Counter = Counter + 1
   Loop While Counter < 32
   If Counter < 32 Then
      daySel = Counter ' Store to global variable.
   End If
End Sub

Private Sub cboDay_Click()
   Dim Counter As Integer
   Counter = 1
   Do
      If (day(Counter) = cboDay.Text) Then
         Exit Do
      End If
      Counter = Counter + 1
   Loop While Counter < 32
   daySel = Counter ' Store to global variable.
End Sub


Private Sub cboMonth_Change()
   Dim Counter As Integer
   Counter = 1
   Do
      If (UCase(month(Counter)) = UCase(cboMonth.Text)) Then
         Exit Do
      End If
      Counter = Counter + 1
   Loop While Counter < 13
   If Counter < 13 Then
      monthSel = Counter ' Store to global variable.
   End If
End Sub
```

```
Private Sub cboMonth_Click()
   Dim Counter As Integer
   Counter = 1
   Do
      If (UCase(month(Counter)) = UCase(cboMonth.Text)) Then
         Exit Do
      End If
      Counter = Counter + 1
   Loop While Counter < 13
   monthSel = Counter ' Store to global variable.
End Sub

Private Sub cboOper_Change()
   addOperation
End Sub

Private Sub cboOper_Click()
   addOperation
End Sub

Private Sub cmdAddOper_Click()
   addOperation
End Sub

Private Sub cmdCancel_Click()
   Dim Counter As Integer
   Dim resp As Integer

   resp = MsgBox("Do you wish to close?", vbYesNo + vbCritical + vbDefaultButton2, "EPIC-View")
   If resp = vbYes Then
      End
   End If
End Sub

Private Sub cmdClose_Click()
   Dim Counter As Integer
   Dim index As String
   Dim nroVal As String
   Dim resp As Integer
   Dim fileName As String

   ' Store the values if a user chooses to.
   resp = MsgBox("Do you wish load this set of management practices and close?", vbYesNo + vbCritical
+ vbDefaultButton2, "EPIC-View")
   If resp = vbYes Then
      Counter = 1
      Do
         cellFiles(Counter) = FreeFile
         fileName = "mgmt" + selectedCells(Counter) + ".utl"
         Open cellDatasetDir + fileName For Append As cellFiles(Counter)
         'Store all management practices related variables (one time only).
         If cellExist(Counter) = 0 Then
            Print #cellFiles(Counter), "NRO   " + Str(nro(Counter))
```

```
            Print #cellFiles(Counter), "NIRR  " + Str(nirr)
            Print #cellFiles(Counter), "IRR   " + Str(irr)
            Print #cellFiles(Counter), "IRI   " + Str(iri)
            Print #cellFiles(Counter), "IFA   " + Str(ifa)
            Print #cellFiles(Counter), "LM    " + Str(lm)
            Print #cellFiles(Counter), "IFD   " + Str(ifd)
            Print #cellFiles(Counter), "IDR   " + Str(idr)
            Print #cellFiles(Counter), "IFFR  " + Str(iffr)
            Print #cellFiles(Counter), "IDFT  " + Str(idft)
            Print #cellFiles(Counter), "BIR   " + Str(bir)
            Print #cellFiles(Counter), "EFI   " + Str(efi)
            Print #cellFiles(Counter), "VIMX  " + Str(vimx)
            Print #cellFiles(Counter), "ARMN  " + Str(armn)
            Print #cellFiles(Counter), "ARMX  " + Str(armx)
            Print #cellFiles(Counter), "BFT   " + Str(bft)
            Print #cellFiles(Counter), "FNP   " + Str(fnp)
            Print #cellFiles(Counter), "FMX   " + Str(fmx)
            Print #cellFiles(Counter), "DRT   " + Str(drt)
            Print #cellFiles(Counter), "FDSF  " + Str(fdsf)
        ElseIf (didCropRotation = 1) Then
            Print #cellFiles(Counter), "NRO   " + Str(nro(Counter))
        End If
        Close cellFiles(Counter)
        cellFiles(Counter) = FreeFile
        ' Rewrite the current number of operations and current NRO value.
        Open cellDatasetDir + fileName For Binary As cellFiles(Counter)
        index = Space(4 - Len(Str(cellIndex(Counter))))
        nroVal = Space(4 - Len(Str(nro(Counter))))
        index = index + Str(cellIndex(Counter))
        nroVal = nroVal + Str(nro(Counter))
        Put #cellFiles(Counter), 4, index
        Put #cellFiles(Counter), 10, nroVal
        Close cellFiles(Counter)
        Counter = Counter + 1
      Loop While Counter <= totCellsSel ' Loop for all selected cells.
      End
    End If
End Sub

Private Sub cmdHelp_Click()
    frmMgmtHelp.Show
End Sub

Private Sub cmdMore_Click()
    frmMgmtDefa.Show
End Sub

Private Sub cmdNewCrop_Click()
    Dim Counter As Integer
    Dim resp As Integer
    ' Provide phenomenon for crop rotation if a user chooses to.
    If justDidCropRotation = 0 Then ' To avoid consicutive crop rotations.
        resp = MsgBox("Do you wish to add new crop rotation?", vbYesNo + vbCritical + vbDefaultButton2,
"EPIC-View")
```

```
    If resp = vbYes Then
      justDidCropRotation = 1
      Counter = 1
      Do
        If (cellIndex(Counter) > 0) Then
          cellIndex(Counter) = cellIndex(Counter) + 1 'Increment no. of operations.
          nro(Counter) = nro(Counter) + 1 'Increment NRO value.
        End If
        Counter = Counter + 1
      Loop While Counter <= totCellsSel 'Loop for all selected cells.
    didCropRotation = 1
    End If
  Else 'If a user attempted consecutive crop rotations.
    resp = MsgBox("Please enter an operation before another crop rotation.", vbInformation, "EPIC-
View")
  End If

End Sub

Private Sub Form_Load()
  Dim Counter As Integer
  Dim fileName As String
  Dim oper As String
  Dim opCode As Integer
  Dim path As String
  Dim str1 As String
  Dim resp As Integer
  Dim totCells As Integer
  Dim genericFlag As String

  'Store months.
  month(1) = "JANUARY"
  month(2) = "FEBRUARY"
  month(3) = "MARCH"
  month(4) = "APRIL"
  month(5) = "MAY"
  month(6) = "JUNE"
  month(7) = "JULY"
  month(8) = "AUGUST"
  month(9) = "SEPTEMBER"
  month(10) = "OCTOBER"
  month(11) = "NOVEMBER"
  month(12) = "DECEMBER"

  'Store units type.
  unitType(0) = "ENGLISH"
  unitType(1) = "METRIC"

  'Create months list.
  Counter = 1
  Do
    cboMonth.AddItem month(Counter)
    Counter = Counter + 1
  Loop While Counter < 13
```

```
cboMonth.ListIndex = 0

'Create days list.
Counter = 1
Do
   day(Counter) = Str(Counter)
   cboDay.AddItem day(Counter)
   Counter = Counter + 1
Loop While Counter < 32
cboDay.ListIndex = 0

'Get directory paths.
fileNum = FreeFile
Open "c:\EVPaths.txt" For Input As fileNum
Input #fileNum, cellDatasetDir, epicOutputDir, soilDir, epicDir
Close fileNum

'Check if this set is generic for whole field.
fileNum = FreeFile
fileName = cellDatasetDir + "selected.cll"
Open fileName For Input As fileNum
Input #fileNum, path, totCells, genericFlag
Counter = 1
totCellsSel = 0
If UCase(genericFlag) = "FALSE" Then 'For selected cells only.
   Do
      If (EOF(fileNum)) Then
         Exit Do
      End If
      Input #fileNum, selectedCells(Counter)
      Counter = Counter + 1
      totCellsSel = totCellsSel + 1
   Loop While Not EOF(fileNum)
   resp = 0
Else 'Generic for the field.
   Do
      selectedCells(Counter) = Counter
      Counter = Counter + 1
      totCellsSel = totCellsSel + 1
   Loop While Counter <= totCells
   resp = vbYes
End If
Close fileNum

'Resize all arrays.
ReDim cellFiles(1 To totCellsSel)
ReDim cellIndex(1 To totCellsSel)
ReDim cellExist(1 To totCellsSel)
ReDim openMode(1 To totCellsSel)
ReDim nro(1 To totCellsSel)

'Initialize NRO array with 1.
Counter = 1
Do
```

```
    nro(Counter) = 1
    Counter = Counter + 1
  Loop While Counter <= totCellsSel
  Counter = 1
  ' Open all selected cells' mgmt#.utl files for writing/appending.
  Do
    fileName = "mgmt" + selectedCells(Counter) + ".utl"
    path = Dir(cellDatasetDir + fileName)
    If path = UCase(fileName) Then
      currentCell = selectedCells(Counter)
      If (resp = 0) Then
        resp = MsgBox("Overwrite all previously existing management files?", vbYesNo + vbQuestion +
vbDefaultButton2, "EPIC-View")
      End If
      If (resp = vbNo) Then
        cellFiles(Counter) = FreeFile
        ' Read the previous index and NRO values.
        Open cellDatasetDir + fileName For Input As cellFiles(Counter)
        str1 = Input(2, cellFiles(Counter))
        Input #cellFiles(Counter), cellIndex(Counter), nro(Counter)
        Close cellFiles(Counter)
        If (cellIndex(Counter) > 0) Then
          cellExist(Counter) = 1
        Else
          cellExist(Counter) = 0
        End If
      ElseIf (resp = vbYes) Then
        cellFiles(Counter) = FreeFile
        Open cellDatasetDir + fileName For Output As cellFiles(Counter)
        Print #cellFiles(Counter), " !    0     1"  'Write initial index on this line."
        Close cellFiles(Counter)
      End If
    Else
      cellFiles(Counter) = FreeFile
      Open cellDatasetDir + fileName For Output As cellFiles(Counter)
      Print #cellFiles(Counter), " !    0     1"  'Write initial index on this line.
      Close cellFiles(Counter)
    End If
    Counter = Counter + 1
  Loop While Counter <= totCellsSel

  'Open the management operation file and create a list of
  'management operations.
  fileNum = FreeFile
  fileName = epicDir + "mgmtoper.dat"
  Open fileName For Input As fileNum
  Counter = 1
  Do
    If (EOF(fileNum)) Then
      Exit Do
    End If
    Input #fileNum, opCode
    operCode(Counter) = opCode
    Input #fileNum, operation(Counter)
```

```
      cboOper.AddItem operation(Counter)
      Counter = Counter + 1
   Loop While Not (EOF(fileNum) Or (Counter > 100))
   firstTime = 0
   cboOper.ListIndex = 0 'Set highlight to first item in list.
   Close fileNum
   initMgmtVars 'Initialize management related variables.
   firstTime = 1
End Sub


' InitMgmtVars
' Author: Anoop Govil
' Date: May 19, 1996
'--------------------------------------------
' initMgmtVars Subroutine:
'  - Initializes all global variables.
'--------------------------------------------
Public Sub initMgmtVars()
   armn = Val(frmMgmtDefa.txtARMN.Text)
   armx = Val(frmMgmtDefa.txtARMX.Text)
   bft = Val(frmMgmtDefa.txtBFT.Text)
   bir = Val(frmMgmtDefa.txtBIR.Text)
   drt = Val(frmMgmtDefa.txtDRT.Text)
   efi = Val(frmMgmtDefa.txtEFI.Text)
   fdsf = Val(frmMgmtDefa.txtFDSF.Text)
   fmx = Val(frmMgmtDefa.txtFMX.Text)
   fnp = Val(frmMgmtDefa.txtFNP.Text)
   idft = Val(frmMgmtDefa.txtIDFT.Text)
   idr = Val(frmMgmtDefa.txtIDR.Text)
   ifa = Val(frmMgmtDefa.txtIFA.Text)
   ifd = Val(frmMgmtDefa.txtIFD.Text)
   iffr = Val(frmMgmtDefa.txtIFFR.Text)
   iri = Val(frmMgmtDefa.txtIRI.Text)
   irr = Val(frmMgmtDefa.txtIRR.Text)
   lm = Val(frmMgmtDefa.txtLM.Text)
   nirr = Val(frmMgmtDefa.txtNIRR.Text)
   didCropRotation = 0
   justDidCropRotation = 0
   vimx = Val(frmMgmtDefa.txtVIMX.Text)
End Sub


' AddOperation
' Author: Anoop Govil
' Date: May 19, 1996
'--------------------------------------------
' addOperation Subroutine:
'  - Checks for the operation code selected by
'  - a user and calls appropriate form to enter
'  - remaining data.
'--------------------------------------------
Public Sub addOperation()
   Dim Counter As Integer
```

```
    Dim resp As Integer
    Dim choice As Integer
    Dim fileName As String

    Counter = 1
    Do
      If (operation(Counter) = UCase(cboOper.Text)) Then
        Exit Do
      End If
      Counter = Counter + 1
    Loop While Counter < 100
    If Counter < 100 Then
      operationCode = operCode(Counter)
      If firstTime <> 0 Then
        choice = operationCode
        If choice = 71 Then ' Fertilize
          justDidCropRotation = 0
          frmFert.Show
        ElseIf choice = 11 Then ' Sprayer.
          justDidCropRotation = 0
          frmPest.Show
        ElseIf choice = 2 Then ' Row Planter.
          justDidCropRotation = 0
          frmRowPlntr.Show
        ElseIf choice = 72 Then ' Irrigation.
          justDidCropRotation = 0
          frmIrrig.Show
        ElseIf (choice = 19 Or choice = 21 Or choice = 23 Or choice = 29 _
            Or choice = 30 Or choice = 51) Then
          justDidCropRotation = 0
          frmCultivate.Show
        ElseIf (choice = 41 Or choice = 28 Or choice = 33) Then
          resp = MsgBox("Do you wish to store this operation?", vbYesNo + vbCritical +
vbDefaultButton2, "EPIC-View")
          If resp = vbYes Then
            justDidCropRotation = 0
            Counter = 1
            Do
              cellFiles(Counter) = FreeFile
              fileName = "mgmt" + selectedCells(Counter) + ".utl"
              Open cellDatasetDir + fileName For Append As cellFiles(Counter)
              cellIndex(Counter) = cellIndex(Counter) + 1
              Print #cellFiles(Counter), "MON(" + Trim(Str(cellIndex(Counter))) + ")  " + Str(monthSel)
              Print #cellFiles(Counter), "DAY(" + Trim(Str(cellIndex(Counter))) + ")  " + Str(daySel)
              Print #cellFiles(Counter), "COD(" + Trim(Str(cellIndex(Counter))) + ")  " +
Str(operationCode)
              Close cellFiles(Counter)
              Counter = Counter + 1
            Loop While Counter <= totCellsSel
          End If
        End If
      End If
    End If
End Sub
```

```
'Mgmtdefa.frm
' This form allows a user to enter management related variables for selected cells or for
'  the whole field as chosen by a user. These are  stored in the cell specific "mgmt.utl" file.

Private Sub cmdHelp_Click()
   frmMgmDefHlp.Show
End Sub

 Private Sub cmdOk_Click()
   'Set global variables.
   armn = Val(txtARMN.Text)
   armx = Val(txtARMX.Text)
   bft = Val(txtBFT.Text)
   bir = Val(txtBIR.Text)
   drt = Val(txtDRT.Text)
   efi = Val(txtEFI.Text)
   fdsf = Val(txtFDSF.Text)
   fmx = Val(txtFMX.Text)
   fnp = Val(txtFNP.Text)
   idft = Val(txtIDFT.Text)
   idr = Val(txtIDR.Text)
   ifa = Val(txtIFA.Text)
   ifd = Val(txtIFD.Text)
   iffr = Val(txtIFFR.Text)
   iri = Val(txtIRI.Text)
   irr = Val(txtIRR.Text)
   lm = Val(txtLM.Text)
   nirr = Val(txtNIRR.Text)
   vimx = Val(txtVIMX.Text)

   Hide
End Sub

Private Sub txtARMN_Change()
   validateText = txtARMN.Text
   validateData
   txtARMN.Text = validateText
End Sub


Private Sub txtARMX_Change()
   validateText = txtARMX.Text
   validateData
   txtARMX.Text = validateText
End Sub


Private Sub txtBFT_Change()
   validateText = txtBFT.Text
   validateData
   txtBFT.Text = validateText
End Sub
```

```
Private Sub txtBIR_Change()
   validateText = txtBIR.Text
   validateData
   txtBIR.Text = validateText
End Sub


Private Sub txtDRT_Change()
   validateText = txtDRT.Text
   validateData
   txtDRT.Text = validateText
End Sub


Private Sub txtEFI_Change()
   validateText = txtEFI.Text
   validateData
   txtEFI.Text = validateText
End Sub


Private Sub txtFDSF_Change()
   validateText = txtFDSF.Text
   validateData
   txtFDSF.Text = validateText
End Sub


Private Sub txtFMX_Change()
   validateText = txtFMX.Text
   validateData
   txtFMX.Text = validateText
End Sub


Private Sub txtFNP_Change()
   validateText = txtFNP.Text
   validateData
   txtFNP.Text = validateText
End Sub


Private Sub txtIDFT_Change()
   validateText = txtIDFT.Text
   validateData
   txtIDFT.Text = validateText
End Sub


Private Sub txtIDR_Change()
   validateText = txtIDR.Text
   validateData
   txtIDR.Text = validateText
End Sub
```

```
Private Sub txtIFA_Change()
   validateText = txtIFA.Text
   validateData
   txtIFA.Text = validateText
End Sub


Private Sub txtIFD_Change()
   validateText = txtIFD.Text
   validateData
   txtIFD.Text = validateText
End Sub


Private Sub txtIFFR_Change()
   validateText = txtIFFR.Text
   validateData
   txtIFFR.Text = validateText
End Sub


Private Sub txtIRI_Change()
   validateText = txtIRI.Text
   validateData
   txtIRI.Text = validateText
End Sub


Private Sub txtIRR_Change()
   validateText = txtIRR.Text
   validateData
   txtIRR.Text = validateText
End Sub


Private Sub txtLM_Change()
   validateText = txtLM.Text
   validateData
   txtLM.Text = validateText
End Sub


Private Sub txtNIRR_Change()
   validateText = txtNIRR.Text
   validateData
   txtNIRR.Text = validateText
End Sub


Private Sub txtVIMX_Change()
   validateText = txtVIMX.Text
   validateData
```

```
    txtVIMX.Text = validateText
End Sub
```

```
'Fert.frm
' This form allows a user to select a fertilizer from a list of fertilizers and other variable
' values related to fertilize operation and store them in the cell specific "mgmt.utl" files.

Option Explicit

Dim fertilizer(1 To 100) As String
Dim fileNum As Integer

Private Sub cmdCancel_Click()
  Hide
End Sub

Private Sub cmdOk_Click()
  Dim Counter As Integer
  Dim ind As Integer
  Dim resp As Integer
  Dim addStr As String
  Dim fileName As String

  'Get the fertilizer code for selected fertilizer.
  ind = 1
  Do
    If fertilizer(ind) = cboFert.Text Then
      Exit Do
    End If
    ind = ind + 1
  Loop While ind < 100

  'Store the management operation if a user chooses to.
  Counter = 1
  If (ind < 100) Then
    resp = MsgBox("Do you wish to store this operation?", vbYesNo + vbCritical + vbDefaultButton2,
"EPIC-View")
    If resp = vbYes Then
      If (UCase(cboUnit.Text) = "ENGLISH") Then
        addStr = "E"
      Else
        addStr = ""
      End If
      Do
        cellFiles(Counter) = FreeFile
        fileName = "mgmt" + selectedCells(Counter) + ".utl"
        Open cellDatasetDir + fileName For Append As cellFiles(Counter)
        cellIndex(Counter) = cellIndex(Counter) + 1
        Print #cellFiles(Counter), "MON(" + Trim(Str(cellIndex(Counter))) + ")  " + Str(monthSel)
        Print #cellFiles(Counter), "DAY(" + Trim(Str(cellIndex(Counter))) + ")  " + Str(daySel)
        Print #cellFiles(Counter), "COD(" + Trim(Str(cellIndex(Counter))) + ")  " + Str(operationCode)
        Print #cellFiles(Counter), "FN(" + Trim(Str(cellIndex(Counter))) + ")  " + Str(ind)
        Print #cellFiles(Counter), "FAP(" + Trim(Str(cellIndex(Counter))) + ")  " + txtAppRate.Text +
addStr
        Print #cellFiles(Counter), "FDP(" + Trim(Str(cellIndex(Counter))) + ")  " + txtFertDepth.Text +
addStr
        Print #cellFiles(Counter), "HUSC(" + Trim(Str(cellIndex(Counter))) + ")  " + txtHUSched.Text
```

```
            Close cellFiles(Counter)
            Counter = Counter + 1
          Loop While Counter <= totCellsSel 'Loop for all selected cells.
        Hide
        End If
    End If
End Sub

Private Sub Form_Load()
    Dim Counter As Integer
    Dim fileName As String
    Dim code As Integer

    'Open fertilizer file and create fertilizer list.
    fileNum = FreeFile
    fileName = epicDir + "fertdata.dat"
    Open fileName For Input As fileNum
    Counter = 1
    Do
      If EOF(fileNum) Then
        Exit Do
      End If
      Input #fileNum, code
      'These codes do not have any operations.
      If ((code < 6 Or code > 10) And (code < 16 Or code > 20) _
          And code <> 25 And (code < 27 Or code > 30) And code <> 35 _
          And code <> 37 And code <> 39 And code <> 40 And (code < 42 Or code > 49)) Then
        Input #fileNum, fertilizer(Counter)
        cboFert.AddItem fertilizer(Counter)
      End If
      Counter = Counter + 1
    Loop While Not (EOF(fileNum) And Counter > 100)
    cboFert.ListIndex = 0
    Close fileNum
    cboUnit.AddItem unitType(0)
    cboUnit.AddItem unitType(1)
    cboUnit.ListIndex = 0
End Sub


Private Sub txtAppRate_Change()
    validateText = txtAppRate.Text
    validateData
    txtAppRate.Text = validateText
End Sub


Private Sub txtFertDepth_Change()
    validateText = txtFertDepth.Text
    validateData
    txtFertDepth.Text = validateText
End Sub
```

```
Private Sub txtHUSched_Change()
   validateText = txtHUSched.Text
   validateData
   txtHUSched.Text = validateText
End Sub
```

```
'frmculvt.frm
' This form allows a user to select enter variable values related to cultivation operation
' and store them in the cell specific "mgmt.utl" files.

Private Sub cmdCancel_Click()
   Hide
End Sub

Private Sub cmdOk_Click()
   Dim Counter As Integer
   Dim resp As Integer
   Dim fileName As String

   'Store the management operation of a user chooses to.
   Counter = 1
   resp = MsgBox("Do you wish to store this operation?", vbYesNo + vbCritical + vbDefaultButton2,
"EPIC-View")
   If resp = vbYes Then
      Do
         cellFiles(Counter) = FreeFile
         fileName = "mgmt" + selectedCells(Counter) + ".utl"
         Open cellDatasetDir + fileName For Append As cellFiles(Counter)
         cellIndex(Counter) = cellIndex(Counter) + 1
         Print #cellFiles(Counter), "MON(" + Trim(Str(cellIndex(Counter))) + ")  " + Str(monthSel)
         Print #cellFiles(Counter), "DAY(" + Trim(Str(cellIndex(Counter))) + ")  " + Str(daySel)
         Print #cellFiles(Counter), "COD(" + Trim(Str(cellIndex(Counter))) + ")  " + Str(operationCode)
         Print #cellFiles(Counter), "HUSC(" + Trim(Str(cellIndex(Counter))) + ")  " + txtHUSC.Text
         Close cellFiles(Counter)
         Counter = Counter + 1
      Loop While Counter <= totCellsSel 'Loop for all selected cells.
   Hide
   End If
End Sub

Private Sub Form_Activate()
   activateForm
End Sub

Private Sub Form_Load()
   activateForm
End Sub



' ActivateForm
' Author: Anoop Govil
' Date: May 22, 1996
'-----------------------------------------------
' activateForm Subroutine:
'  - Activates the form with a particular title
'  - so that same form can be used for more than
'  - one management operation.
'-----------------------------------------------
Public Sub activateForm()
```

```
  If operationCode = 19 Then
    frmCultivate.Caption = "EPIC-View - Row Cultivator"
  ElseIf operationCode = 21 Then
    frmCultivate.Caption = "EPIC-View - Hoe"
  ElseIf operationCode = 23 Then
    frmCultivate.Caption = "EPIC-View - Sweep"
  ElseIf operationCode = 29 Then
    frmCultivate.Caption = "EPIC-View - Disk"
  ElseIf operationCode = 30 Then
    frmCultivate.Caption = "EPIC-View - Chisel"
  ElseIf operationCode = 51 Then
    frmCultivate.Caption = "EPIC-View - Harvest"
  End If
End Sub

Private Sub txtHUSC_Change()
  validateText = txtHUSC.Text
  validateData
  txtHUSC.Text = validateText
End Sub
```

```
'Irrig.frm
' This form allows a user to enter values for variables related to irrigate operation and
' store the values in the cell specific "mgmt.utl" files.

Option Explicit

Private Sub cmdCancel_Click()
   Hide
End Sub

Private Sub cmdOk_Click()
   Dim Counter As Integer
   Dim resp As Integer
   Dim addStr As String
   Dim fileName As String

   'Store the operation if a user chooses to.
   Counter = 1
   resp = MsgBox("Do you wish to store this operation?", vbYesNo + vbCritical + vbDefaultButton2,
"EPIC-View")
   If resp = vbYes Then
      If (UCase(cboUnit.Text) = "ENGLISH") Then
         addStr = "E"
      Else
         addStr = ""
      End If
      Do
         cellFiles(Counter) = FreeFile
         fileName = "mgmt" + selectedCells(Counter) + ".utl"
         Open cellDatasetDir + fileName For Append As cellFiles(Counter)
         cellIndex(Counter) = cellIndex(Counter) + 1
         Print #cellFiles(Counter), "MON(" + Trim(Str(cellIndex(Counter))) + ")  " + Str(monthSel)
         Print #cellFiles(Counter), "DAY(" + Trim(Str(cellIndex(Counter))) + ")  " + Str(daySel)
         Print #cellFiles(Counter), "COD(" + Trim(Str(cellIndex(Counter))) + ")  " + Str(operationCode)
         Print #cellFiles(Counter), "IA(" + Trim(Str(cellIndex(Counter))) + ")  " + txtIA.Text + addStr
         Print #cellFiles(Counter), "QVOL(" + Trim(Str(cellIndex(Counter))) + ")  " + txtQVol.Text
         Close cellFiles(Counter)
         Counter = Counter + 1
      Loop While Counter <= totCellsSel 'Loop for all selected cells.
   Hide
   End If
End Sub


Private Sub Form_Load()
   cboUnit.AddItem unitType(0)
   cboUnit.AddItem unitType(1)
   cboUnit.ListIndex = 0
End Sub


Private Sub txtIA_Change()
   validateText = txtIA.Text
   validateData
```

```
    txtIA.Text = validateText
End Sub


Private Sub txtQVol_Change()
    validateText = txtQVol.Text
    validateData
    txtQVol.Text = validateText
End Sub
```

```
'Pest.frm
' This form allows a user to select a pesticide from a list of pesticides and other variable
' values related to sprayer operation and store them in the cell specific "mgmt.utl" files.

Option Explicit

Dim pesticide(1 To 300) As String
Dim fileNum As Integer

Private Sub cmdCancel_Click()
   Hide
End Sub

Private Sub cmdOk_Click()
   Dim Counter As Integer
   Dim ind As Integer
   Dim resp As Integer
   Dim addStr As String
   Dim fileName As String

   'Get the pesticide code.
   ind = 1
   Do
      If pesticide(ind) = cboPest.Text Then
         Exit Do
      End If
      ind = ind + 1
   Loop While ind < 100
   Counter = 1
   If (ind < 100) Then
      'Store the operation if a user chooses to.
      resp = MsgBox("Do you wish to store this operation?", vbYesNo + vbCritical + vbDefaultButton2,
"EPIC-View")
      If resp = vbYes Then
        If (UCase(cboUnit.Text) = "ENGLISH") Then
           addStr = "E"
        Else
           addStr = ""
        End If
        Do
           cellFiles(Counter) = FreeFile
           fileName = "mgmt" + selectedCells(Counter) + ".utl"
           Open cellDatasetDir + fileName For Append As cellFiles(Counter)
           cellIndex(Counter) = cellIndex(Counter) + 1
           Print #cellFiles(Counter), "MON(" + Trim(str(cellIndex(Counter))) + ")  " + str(monthSel)
           Print #cellFiles(Counter), "DAY(" + Trim(str(cellIndex(Counter))) + ")  " + str(daySel)
           Print #cellFiles(Counter), "COD(" + Trim(str(cellIndex(Counter))) + ")  " + str(operationCode)
           Print #cellFiles(Counter), "PST(" + Trim(str(cellIndex(Counter))) + ")  " + str(ind)
           Print #cellFiles(Counter), "PCF(" + Trim(str(cellIndex(Counter))) + ")  " + txtPCF.Text
           Print #cellFiles(Counter), "PAR(" + Trim(str(cellIndex(Counter))) + ")  " + txtAppRate.Text +
addStr
           Close cellFiles(Counter)
           Counter = Counter + 1
        Loop While Counter <= totCellsSel 'Loop for all selected cells.
```

```
      Hide
      End If
   End If
End Sub

Private Sub Form_Load()
   Dim Counter As Integer
   Dim fileName As String
   Dim pest As String
   Dim str As String

   'Open pesticide file and create a list of pesticides.
   fileNum = FreeFile
   fileName = epicDir + "usdapest.dat"
   Open fileName For Input As fileNum
   Counter = 1
   Do
      If EOF(fileNum) Then
         Exit Do
      End If
      pest = Input(16, fileNum)
      pesticide(Counter) = pest
      cboPest.AddItem pesticide(Counter)
      Input #fileNum, str
      Counter = Counter + 1
   Loop While Not (EOF(fileNum) And Counter > 300)
   cboPest.ListIndex = 0
   Close fileNum
   cboUnit.AddItem unitType(0)
   cboUnit.AddItem unitType(1)
   cboUnit.ListIndex = 0
End Sub


Private Sub txtAppRate_Change()
   validateText = txtAppRate.Text
   validateData
   txtAppRate.Text = validateText
End Sub


Private Sub txtPCF_Change()
   validateText = txtPCF.Text
   validateData
   txtPCF.Text = validateText
End Sub
```

```
'Rowplntr.frm
' This form allows a user to select a crop from a list of crops and other variable
' values related to rowplanter operation and store them in the cell specific "mgmt.utl" file.

Option Explicit

Dim Crop(1 To 100) As String
Dim fileNum As Integer

Private Sub cmdCancel_Click()
  Hide
End Sub

Private Sub cmdOk_Click()
  Dim Counter As Integer
  Dim ind As Integer
  Dim resp As Integer
  Dim fileName As String

  'Get the crop code.
  ind = 1
  Do
    If Crop(ind) = cboCrop.Text Then
      Exit Do
    End If
    ind = ind + 1
  Loop While ind < 100

  'Store the operation if a user chooses to.
  Counter = 1
  If (ind < 100) Then
    resp = MsgBox("Do you wish to store this operation?", vbYesNo + vbCritical + vbDefaultButton2,
"EPIC-View")
    If resp = vbYes Then
      Do
        cellFiles(Counter) = FreeFile
        fileName = "mgmt" + selectedCells(Counter) + ".utl"
        Open cellDatasetDir + fileName For Append As cellFiles(Counter)
        cellIndex(Counter) = cellIndex(Counter) + 1
        Print #cellFiles(Counter), "MON(" + Trim(Str(cellIndex(Counter))) + ")  " + Str(monthSel)
        Print #cellFiles(Counter), "DAY(" + Trim(Str(cellIndex(Counter))) + ")  " + Str(daySel)
        Print #cellFiles(Counter), "COD(" + Trim(Str(cellIndex(Counter))) + ")  " + Str(operationCode)
        Print #cellFiles(Counter), "CRP(" + Trim(Str(cellIndex(Counter))) + ")  " + Str(ind)
        Print #cellFiles(Counter), "GRZ(" + Trim(Str(cellIndex(Counter))) + ")  " + Str(ind)
        Print #cellFiles(Counter), "PHU(" + Trim(Str(cellIndex(Counter))) + ")  " + txtPHU.Text
        Close cellFiles(Counter)
        Counter = Counter + 1
      Loop While Counter <= totCellsSel 'Loop for all selected cells.
      Hide
    End If
  End If
End Sub

Private Sub Form_Load()
```

```
    Dim Counter As Integer
    Dim fileName As String
    Dim code As Integer
    Dim crpCode As String
    Dim crp As String

    'Open the crop file and create the crops list.
    fileNum = FreeFile
    fileName = epicDir + "usdacrop.txt"
    Open fileName For Input As fileNum
    Counter = 1
    Do
      If EOF(fileNum) Then
        Exit Do
      End If
      Input #fileNum, code
      If (code <> 9 And code <> 29) Then
        crpCode = Input(4, fileNum)
        Input #fileNum, crp
        Crop(Counter) = crp
        cboCrop.AddItem Crop(Counter)
      Else
        Input #fileNum, crp
      End If
      Counter = Counter + 1
    Loop While Not (EOF(fileNum) And Counter > 100)
    cboCrop.ListIndex = 0
    Close fileNum

End Sub


Private Sub txtPHU_Change()
    validateText = txtPHU.Text
    validateData
    txtPHU.Text = validateText
End Sub
```

```
'Outputop.frm
' This form allows a user to select a list of variables to be monitored as a result of running
' EPIC on the selected cells. These can be selected from a list of output variables
' provided. Also a user can opt to select daily, monthly, yearly, annual or all these output
' files to be generated by EPIC. These are stored into "prnt.utl" file which are later loaded
' into "prnt5300.dat" file.

Option Explicit

Dim fileNum As Integer
Dim cellDatasetDir As String
Dim epicOutputDir As String
Dim soilDir As String
Dim epicDir As String
Dim outputVars(1 To 150) As String
Dim totVarsSel As Integer


Private Sub cmdCancel_Click()
  Dim resp As Integer

  resp = MsgBox("Do you wish to close?", vbYesNo + vbQuestion, "EPIC-View")
  If resp = vbYes Then
    End
  End If
End Sub

Private Sub cmdOk_Click()
  Dim Counter As Integer
  Dim ctr As Integer
  Dim resp As Integer
  Dim outputStr As String

  Counter = 0
  totVarsSel = 1
  outputStr = ""
  'Restore old settings/store new settings as choosen by a user.
  If (lstOpVars.SelCount > 0 Or optOldVal.Value = True) Then
    resp = MsgBox("Do you wish to close?", vbYesNo + vbQuestion, "EPIC-View")
    If resp = vbYes Then
      If optNewVal.Value = True Then
        fileNum = FreeFile
        'Writing a string for output files depending upon user choice.
        Open cellDatasetDir + "outfiles.dat" For Output As fileNum
        If Not (optDaily.Value = False And optMonthly.Value = False And optYearly = False _
            And optAnnual.Value = False And optAllFiles.Value = False) Then
          If optDaily.Value = True Then 'For daily output.
            outputStr = " -epd "
          ElseIf optMonthly.Value = True Then 'For monthly output.
            outputStr = " -epm "
          ElseIf optYearly.Value = True Then 'For yearly output.
            outputStr = " -epy "
          ElseIf optAnnual.Value = True Then 'For annual output.
            outputStr = " -epa "
```

```
            ElseIf optAllFiles.Value = True Then
              outputStr = " -ep " 'For all above outputs.
            End If
            Print #fileNum, outputStr
          Else 'For none of above outputs.
            Print #fileNum, "NONE"
          End If
          Close fileNum
          fileNum = FreeFile
          'Creating the output variables file.
          Open cellDatasetDir + "prnt.utl" For Output As fileNum
          'Writing for default values of daily and monthly outputs.
          Print #fileNum, "KD(1)    0"
          Print #fileNum, "KM(1)    0"
          'Writing the codes for all output variables selected.
          Do
          If lstOpVars.Selected(Counter) = True Then
            ctr = 1
            'Loop to get the variable's code.
            Do
              If outputVars(ctr) = lstOpVars.List(Counter) Then
                Exit Do
              End If
              ctr = ctr + 1
            Loop While ctr < 150
            If ctr < 150 Then
              Print #fileNum, "KY(" + Trim(Str(totVarsSel)) + ")   " + Str(ctr)
              totVarsSel = totVarsSel + 1
            End If
          End If
          Counter = Counter + 1
          Loop While (Counter < lstOpVars.ListCount And totVarsSel < 30)
          'If less than 30 variables were selected, write '0' for all
          ' remaining variable places.
          If totVarsSel < 30 Then
            Do
              Print #fileNum, "KY(" + Trim(Str(totVarsSel)) + ")   0"
              totVarsSel = totVarsSel + 1
            Loop While totVarsSel <= 30
          End If
          Close fileNum
        End If
        End
      End If
    Else 'In case, no variable is selected.
      If optNewVal.Value = True Then
        resp = MsgBox("The output variables are not selected!", vbInformation, "EPIC-View")
      End If
    End If
End Sub

Private Sub Form_Load()

  Dim Counter As Integer
```

```
  Dim fileName As String
  Dim ctr As Integer

  'Get directory paths.
  fileNum = FreeFile
  Open "c:\EVPaths.txt" For Input As fileNum
  Input #fileNum, cellDatasetDir, epicOutputDir, soilDir, epicDir
  Close fileNum

  'Open output variables file and create a list of output variables.
  fileNum = FreeFile
  fileName = epicDir + "opvarlst.dat"
  Open fileName For Input As fileNum
  Counter = 1
  totVarsSel = 0
  txtTotSel.Text = Str(lstOpVars.SelCount)
  Do
    If EOF(fileNum) Then
      Exit Do
    End If
    Input #fileNum, ctr, outputVars(Counter)
    lstOpVars.AddItem outputVars(Counter)
    Counter = Counter + 1
  Loop While Not (EOF(fileNum) And Counter > 150)
  Close fileNum

End Sub


Private Sub lstOpVars_Click()
    Dim resp As Integer

    txtTotSel.Text = (lstOpVars.SelCount)
    If lstOpVars.SelCount > 30 Then
      resp = MsgBox("More than 30 output variable(s) have been selected! Last selected variable(s) will
be ignored.", vbCritical, "EPIC-View")
    End If
End Sub


Private Sub lstOpVars_DblClick()
  Dim resp As Integer

  txtTotSel.Text = (lstOpVars.SelCount)
  If lstOpVars.SelCount > 30 Then
    resp = MsgBox("More than 30 output variable(s) have been selected! Last selected variable(s) will be
ignored.", vbCritical, "EPIC-View")
  End If
End Sub


Private Sub optNewVal_Click()
  optDaily.Enabled = True
  optMonthly.Enabled = True
```

```
    optYearly.Enabled = True
    optAnnual.Enabled = True
    optAllFiles.Enabled = True
    lblOpVars.Enabled = True
    lblTotSel.Enabled = True
    lstOpVars.Enabled = True
    Frame2.Enabled = True
End Sub


Private Sub optOldVal_Click()
    optDaily.Enabled = False
    optMonthly.Enabled = False
    optYearly.Enabled = False
    optAnnual.Enabled = False
    optAllFiles.Enabled = False
    lblOpVars.Enabled = False
    lblTotSel.Enabled = False
    lstOpVars.Enabled = False
    Frame2.Enabled = False
End Sub
```

APPENDIX G

AVENUE® CODE FOR INTERFACING

```
' Epic.constantData
' This Script writes total number of cells, available in the gridded
' coverage, to file "selected.cll" and then invokes the constant data
' entry user interface.
' Prepared by Anoop Govil
' Dated 5/19/96
'
selectedCell=(_cellDatasetDir.AsString+"selected.cll").AsFileName
selectedFile = TextFile.Make(selectedCell, #FILE_PERM_WRITE)
selectedFile.Write(_totalCells.AsString, _totalCells.AsString.Count)
selectedFile.WriteElt(_newLineChar)
selectedFile.Close
_epicDir.AsFileName.setCWD
command = _exeDir+"constdat.exe"
system.execute(command)
' Disable the Constant Data menu option.
_constEnableFlag = 0


' Epic.DispChart
' This Script displays Chart as per user choices.
' Prepared by Anoop Govil
' Dated 5/19/96
'
' Create an output variables list.
outputVarList = List.Make
theTable = av.GetProject.FindDoc(_mainTable)
resTable = av.GetProject.FindDoc(_resultsTable)
for each aField in resTable.GetVTab.GetFields
  if( (aField.GetAlias <> _cellIdFld) And (aField.GetAlias <> "")
      And (aField.GetAlias <> "Years") And (aField.GetAlias <> "Botch.da") )then
    outputVarList.Add(aField.GetAlias)
  end
end
'for each item in outputVarList
'  MsgBox.Info(item, "EPIC-View")
'end
'userList = MsgBox.MultiInput("", "EPIC-View",
'
' Display the chart properties option.
'
aChart = Chart.MakeUsingDialog(resTable.GetVTab)
if(aChart <> nil) then
  aChart.GetWin.Open
end
```

```
' Epic.displayMap
' This Script loads the results table, created as a result of parsing the
' EPIC output, to the project, joins it with the main theme's attribute
' table and creates new themes depending upon the output variables selected
' by a user and displays the themes in the current field view in different
' colors.
' Prepared by Anoop Govil
' Dated 5/14/96
'
'
' Add the comma delemited file created by parser as a new table in the project and join it with
' the main attribute table on Cell Id.
'
theTable = av.GetProject.FindDoc(_mainTable)
theTableWin=theTable.GetWin
'
' Removes any fields joined to the current table
'
theVTab = theTable.GetVTab
if (theVTab.IsBase.Not) then
  av.GetProject.SetModified(true)
end
theVTab.UnjoinAll
'
resTable = av.GetProject.FindDoc(_resultsTable)
if(resTable = nil) then
  f=(_epicOutputDir+_resultsTable).AsFileName
  v = VTab.Make(f, FALSE, FALSE)
  if (v.HasError) then
    MsgBox.Error("The file '" + f.GetBaseName + "' is not valid.", "")
  else
    t = Table.Make(v)
    t.SetName(v.GetName)
    tField = t.GetVTab.FindField(_cellIdFld)
    t.SetActiveField(tField)
    tableField = theTable.GetVTab.FindField(_cellIdFld)
    theTable.SetActiveField(tableField)
    theTable.GetVTab.Join(tableField,t.GetVTab,tField)
  end
else
    resField = resTable.GetVTab.FindField(_cellIdFld)
    resTable.SetActiveField(resField)
    tableField = theTable.GetVTab.FindField(_cellIdFld)
    theTable.SetActiveField(tableField)
    theTable.GetVTab.Join(tableField,resTable.GetVTab,resField)
end
'
' Replicate main theme into new themes depending upon
' output variables selected by a user.
'
epicProject=av.getProject
fieldView=epicProject.FindDoc(_mainView)
'
' To make sure that only main theme is active.
```

```
'
for each aTheme in fieldView.GetThemes
  if(aTheme.GetName = _mainTheme) then
    aTheme.SetActive(True)
  else
    aTheme.SetActive(False)
  end
end
fieldView.CopyThemes
'
'
resTable = av.GetProject.FindDoc(_resultsTable)
' Initialize theme colors.
r1 = 200
g1 = 200
b1 = 250
r2 = 250
g2 = 150
b2 = 150
for each aField in theTable.GetVTab.GetFields
  fieldExists = resTable.GetVTab.FindField(aField.GetAlias)
  if( (fieldExists <> nil) And (fieldExists.GetAlias <> _cellIdFld) And (fieldExists.GetAlias <> "")
      And (fieldExists.GetAlias <> "Years") And (fieldExists.GetAlias <> "field") )then
    fieldView.Paste
    for each aTheme in fieldView.GetThemes
      if(aTheme.GetName = _mainTheme) then
        aTheme.SetActive(True)
      else
        aTheme.SetActive(False)
      end
    end
    resultTheme = fieldView.FindTheme(_mainTheme)
    resultTheme.SetName(aField.GetAlias)
    resultLegend = resultTheme.GetLegend
    resField = resultTheme.GetFTab.FindField(aField.GetAlias)
    resultLegend.Interval(resultTheme.GetFTab, resField, 5)
'   resultLegend.Quantile(resultTheme.GetFTab, resField, 5)
    resultLegend.SetField(resField)
    startColor = Color.Make
    endColor = Color.Make
    startColor.SetRgbList({r1, g1, b1})  '200, 200, 250
    endColor.SetRgbList({r2, g2, b2})    '250, 150, 150
    resultLegend.RampColors(startColor, endColor)
    ' Change colors for the next theme.
    r1 = r1 + 30
    b1 = b1 + 20
    g1 = g1 + 10
    if (r1 > 255) then
      r1 = 0 + (r1 - 255)
    end
    if (b1 > 255) then
      b1 = 0 + (b1 - 255)
    end
    if (g1 > 255) then
```

```
    g1 = 0 + (g1 - 255)
   end
   r2 = r2 + 70
   b2 = b2 + 60
   g2 = g2 + 50
   if (r2 > 255) then
    r2 = 0 + (r2 - 255)
   end
   if (b2 > 255) then
    b2 = 0 + (b2 - 255)
   end
   if (g2 > 255) then
    g2 = 0 + (g2 - 255)
   end
   resultTheme.SetVisible(False)
   if (resultTheme.Is( FTHEME )) then
    sel = resultTheme.GetFTab.GetSelection
    sel.ClearAll 'Clear all selections of added themes.
    resultTheme.GetFTab.UpdateSelection
   end
  end
end


' Epic.displayTable
' Opens and displays the results table.
' Prepared by Anoop Govil
' Dated 5/23/96
'
epicProject=av.getProject
resTable=epicProject.FindDoc(_resultsTable)
resWin=resTable.GetWin
if (resWin.IsOpen.Not)then
 resWin.Open
else
 resWin.Activate
end
```

```
' Epic.getGISData
' This script retrieves elev, slope, crop, soil series
' field values of the selected records from the theme's table
' Prepared by Anoop Govil
' Dated 2/27/96
'
' Reset any previously existing selectCells list.
totFiles=_selectCellsList.Count
index = totFiles - 1
while (index >= 0)
  _selectCellsList.Remove(index)
  index = index - 1
end
theTable = av.GetProject.FindDoc(_mainTable)
if(nil=theTable)then
  MsgBox.Error("The table: "+_mainTable+", not found.", "Epic")
  exit
end
theTableWin=theTable.GetWin
if (theTableWin.IsOpen.Not)then
  theTableWin.Open
else
  theTableWin.Activate
end
theTableWin.Minimize
'
theVTab = theTable.GetVTab
myVTab = theVTab.GetSelection
if (0=theVTab.GetSelection.Count) then
  MsgBox.Error("There are no cells selected to extract spatial attributes.","EPIC-View")
  exit
end
'
'
'
soilField = theVTab.FindField("Series")
sortField = theVTab.FindField(_cellIdFld)
theTable.Sort(sortField, False)
'
'Show status bar
av.ShowMsg("creating files...")
canceled = False
'av.ShowStopButton
statusIndex = 0
'av.SetStatus (statusIndex)
selRecords=theVTab.GetNumSelRecords
statusIncrement = 100 / selRecords
'
for each rec in myVTab
  cellIdField = theVTab.FindField(_cellIdFld)
  cellId = theVTab.ReturnValueString(cellIdField, rec)
  aFileName=(_cellDatasetDir+"form"+cellId+".utl").AsFileName
  aTextFile = TextFile.Make(aFileName, #FILE_PERM_WRITE)
  _selectCellsList.Add(cellId)
```

```
'Get soil series
if(_userSoilIsAbsent = False)then
  soilSeries = theVTab.ReturnValueString(soilField, rec)
  soilSeries = "@"+_soilDir+soilSeries+".utl"
  aTextFile.Write(soilSeries, soilSeries.Count)
  aTextFile.WriteElt(_newLineChar)
end

'Get elevation value
elevField = theVTab.FindField("Elev")
elev = theVTab.ReturnValueNumber(elevField, rec)
elev = "ELEV      "+elev.AsString
'Write to the text file
aTextFile.Write(elev.AsString, elev.AsString.Count)
aTextFile.WriteElt(_newLineChar)

'Get slope value
slopeField = theVTab.FindField("Slope")
slope = theVTab.ReturnValueNumber(slopeField, rec)
slope = slope / 100
slope="S         "+slope.AsString
'Write to the text file
aTextFile.Write(slope, slope.AsString.Count)
aTextFile.WriteElt(_newLineChar)

'Get area value
areaField = theVTab.FindField("Area")
area = theVTab.ReturnValueNumber(areaField, rec)
area = area / 10000
areaStr="WSA      "+area.AsString
'Write to the text file
aTextFile.Write(areaStr, areaStr.AsString.Count)
aTextFile.WriteElt(_newLineChar)

'Get Runoff Curve number value(if user specified
'soil is absent).
if(_userSoilIsAbsent = False)then
  cNumField = theVTab.FindField("Cn2")
  cNum = theVTab.ReturnValueNumber(cNumField, rec)
  cNum="CN2      "+cNum.AsString
  'Write to the text file
  aTextFile.Write(cNum, cNum.AsString.Count)
  aTextFile.WriteElt(_newLineChar)
end

'Store slope length after calculating it.
area = area * 10000
side = area.Sqrt
SL = side * 2.Sqrt
SLStr = "SL       " + SL.AsString
aTextFile.Write(SLStr, SLStr.Count)
aTextFile.WriteElt(_newLineChar)
```

```
  'Store pointer to cell's corresponding mgmt file.
  mgmtFile = "@"+_cellDatasetDir+"mgmt"+cellId+".utl"
  aTextFile.Write(mgmtFile, mgmtFile.Count)
  aTextFile.WriteElt(_newLineChar)

  aTextFile.Close
'
'  statusIndex = statusIndex + statusIncrement
'  continued = av.SetStatus (statusIndex)
'  if(Not continued) then
'    canceled = true
'    break
'  end
'
end
'
if(canceled) then
  av.ShowMsg("Process interrupted.")
else
  MsgBox.Info("Extracted Spatial data from selected cells.", "Epic")
end
'
' Enable the Output Options and Run Simulator menu options.
_opOptionEnableFlag = 1
_runEpicEnableFlag = True
'


' Epic.mgmtData
' This Script provides a user with choices to make a set of management practices
' generic or select specific cells and enter management practices for
' those cells by invoking Management Practices data entry user interface.
' Prepared by Anoop Govil
' Dated 5/23/96
'
'
' Reset the selectedCells list.
'
totFiles=_selectCellsList.Count
index = totFiles - 1
while (index >= 0)
  _selectCellsList.Remove(index)
  index = index - 1
end
theTable = av.GetProject.FindDoc(_mainTable)
if(nil=theTable)then
  MsgBox.Error("The table: "+_mainTable+", not found.", "Epic")
  exit
end
theTableWin=theTable.GetWin
if (theTableWin.IsOpen.Not)then
  theTableWin.Open
else
  theTableWin.Activate
```

```
end
theTableWin.Minimize
'
genericFlag = false
theVTab = theTable.GetVTab
myVTab = theVTab.GetSelection
'
' If no cells are selected, give user a choice to make management practices generic.
'
if (0=theVTab.GetSelection.Count) then
  genericFlag = MsgBox.YesNo("Do you wish to make this set of management practices generic for the
whole field?", "EPIC-View", False )
  if(genericFlag.Not)then
    MsgBox.Error("In that case, please select the cells for entering management practices.","EPIC-View")
    exit
  end
end
'
sortField = theVTab.FindField(_cellIdFld)
theTable.Sort(sortField, False)
'
' If cells are selected, build the selectedCells list.
'
for each rec in myVTab
  cellIdField = theVTab.FindField(_cellIdFld)
  cellId = theVTab.ReturnValueString(cellIdField, rec)
  _selectCellsList.Add(cellId)
end
'
' Write selected cells ids to file selected.cll.
'
selectedCell=(_cellDatasetDir.AsString+"selected.cll").AsFileName
selectedFile = TextFile.Make(selectedCell, #FILE_PERM_WRITE)
path=_epicOutputDir
selectedFile.Write(path, path.Count)
selectedFile.WriteElt(_newLineChar)
selectedFile.Write(_totalCells.AsString, _totalCells.AsString.Count)
selectedFile.WriteElt(_newLineChar)
selectedFile.Write(genericFlag.AsString, genericFlag.AsString.Count)
selectedFile.WriteElt(_newLineChar)

if (genericFlag.Not) then
  for each cellId in _selectCellsList
    selectedFile.Write(cellId, cellId.Count)
    selectedFile.WriteElt(_newLineChar)
  end
end
selectedFile.Close
'Run Visual Basic management data entry screen.
command = _exeDir+"mgmt.exe"
system.execute(command)
'
```

```
' Epic.outputOptions
' This Script allows a user to enter output options to be monitored
' by providing an Output Options data entry user interface.
' Prepared by Anoop Govil
' Dated 5/24/96
'
'
command = _exeDir+"outputop.exe"
system.execute(command)
' Disable Output Options menu option and enable Run Simulator menu option.
_opOptionEnableFlag = 0
_runEpicEnableFlag = 1



' Epic.removeThemes
' This Script removes the added themes (except the main theme), if
' a user chooses to do so.
' Prepared by Anoop Govil
' Dated 5/12/96
'
epicProject=av.getProject
fieldView=epicProject.FindDoc(_mainView)
if(fieldView.GetThemes.Count > 1) then
 doIt = MsgBox.YesNo("Do you wish to delete all the added themes?", "EPIC-View", TRUE)
 if(doIt) then
  totalThemes = fieldView.GetThemes.Count
  while(totalThemes > 1)
   themesList = fieldView.GetThemes
   for each aTheme in themesList
    if(aTheme.GetName <> _mainTheme) then
     fieldView.DeleteTheme(aTheme)
     totalThemes = totalThemes - 1
     break
    end
   end
  end
 end
end
' Activate main theme.
for each aTheme in fieldView.GetThemes
 if(aTheme.GetName = _mainTheme) then
  aTheme.SetActive(True)
 else
  aTheme.SetActive(False)
 end
end
```

```
' Epic.runEpic
' This Script creates various batch files to complete the cell specifie
' input datasets and then invoke EPIC on all of selected cells' input
' datasets and finally invoke the parser to create a comma delimited
' file from the EPIC output files.
' Prepared by Anoop Govil
' Dated 2/28/96
'
'
' Check if any cells are selected to run EPIC.
'
cellsSelected=_selectCellsList.Count
if(0=cellsSelected)then
 MsgBox.Error("There are no cells selected to run the model.", "EPIC-View")
 exit
end
'
theTable = av.GetProject.FindDoc(_mainTable)
theTableWin=theTable.GetWin
theVTab = theTable.GetVTab
myVTab = theVTab.GetSelection
if (0=theVTab.GetSelection.Count) then
 MsgBox.Error("There are no cells selected.", "EPIC-View")
 exit
end
'
' Removes any fields joined to the current table
'
if (theVTab.IsBase.Not) then
 av.GetProject.SetModified(true)
end
theVTab.UnjoinAll
'
'Removes the added table "form.prs" from project.
'
theProject = av.GetProject
theTable = theProject.FindDoc(_resultsTable)
if(nil <> theTable) then
 theProject.RemoveDoc(theTable)
end
'
' Removes the added themes.
'
epicProject=av.getProject
fieldView=epicProject.FindDoc(_mainView)
if(fieldView.GetThemes.Count > 1) then
   totalThemes = fieldView.GetThemes.Count
   while(totalThemes > 1)
    themesList = fieldView.GetThemes
    for each aTheme in themesList
     if(aTheme.GetName <> _mainTheme) then
      fieldView.DeleteTheme(aTheme)
      totalThemes = totalThemes - 1
      break
```

```
      end
    end
  end
end
'
' If a user selected to have daily, monthly, yearly, annual of all of these
' EPIC output files, a string is written to a file "outfiles.dat" which
' is read here and appropriate command to run EPIC is formulated.
'
userString=""
userPreference=(_cellDatasetDir+"outfiles.dat").AsFileName
if(File.Exists(userPreference)) then
 stringFile=TextFile.Make(userPreference, #FILE_PERM_READ)
 while(stringFile.IsAtEnd.Not)
  listChar = stringFile.ReadElt
  if(listChar = 10.AsChar.AsString) then
   break
  else
   userString = userString + listChar
  end
 end
 if (userString = "NONE") then ' If no output file is required.
  userString=""
 end
end
'
' Creating various batch files for running.
'
makeDataset=(_cellDatasetDir.AsString+"create.bat").AsFileName
createFile = TextFile.Make(makeDataset, #FILE_PERM_WRITE)
runEpic=(_cellDatasetDir.AsString+"runepic.bat").AsFileName
runEpicFile = TextFile.Make(runEpic, #FILE_PERM_WRITE)
selectedCell=(_cellDatasetDir.AsString+"selected.cll").AsFileName
selectedFile = TextFile.Make(selectedCell, #FILE_PERM_WRITE)
path=_epicOutputDir
selectedFile.Write(path, path.Count)
selectedFile.WriteElt(_newLineChar)
'
'Show status bar
av.ShowMsg("creating files...")
canceled = False
av.ShowStopButton
statusIndex = 0
av.SetStatus (statusIndex)
totFiles=_selectCellsList.Count
statusIncrement = 100 / totFiles
'
' Writing commands in the batch files.
'
for each cellId in _selectCellsList
 command=_epicDir+"ewq "+userString+_cellDatasetDir+"form"+cellId+" "+_epicOutputDir
 selectedFile.Write(cellId, cellId.Count)
 selectedFile.WriteElt(_newLineChar)
 runEpicFile.Write(command, command.Count)
```

```
  runEpicFile.WriteElt(_newLineChar)
  command=_epicDir+"util epic "+_cellDatasetDir+"form"+cellId+".dat @"+
        _cellDatasetDir+"form"+cellId+".utl"
  createFile.Write(command, command.Count)
  createFile.WriteElt(_newLineChar)
'
  statusIndex = statusIndex + statusIncrement
  continued = av.SetStatus (statusIndex)
  if(Not continued) then
    canceled = true
    break
  end
'
end
'
if(canceled) then
  av.ShowMsg("Process interrupted.")
else
  av.ShowMsg("Created command file for running simulator.")
end
'
outputOptionString=(_epicDir+"util prnt prnt5300.dat @"+_cellDatasetDir+"prnt.utl").AsFileName
if (File.Exists((_cellDatasetDir+"prnt.utl").AsFileName)) then
  createFile.Write(outputOptionString.AsString, outputOptionString.AsString.Count)
  createFile.WriteElt(_newLineChar)
end
createFile.Write(runEpic.AsString, runEpic.AsString.Count)
createFile.WriteElt(_newLineChar)
createFile.Close
command=_exeDir+"parse.exe"
runEpicFile.Write(command, command.Count)
runEpicFile.WriteElt(_newLineChar)
runEpicFile.Close
selectedFile.Close
_epicDir.AsFileName.setCWD
'
' Invoking a waitshell to run EPIC.
'
command = _exeDir+"DSETMAKE.EXE Beavis Epic.returnToAV"
system.execute(command)
' Disabling Run Simulator menu option.
_runEpicEnableFlag = False
' Enabling the Display Map menu option.
_displayEnableFlag=True
```

```
' Epic.showExtent
' This script opent the main view and activates the main theme.
' Prepared by Anoop Govil
' Dated 2/24/96
'
epicProject=av.getProject
fieldView=epicProject.FindDoc(_mainView)
'
'
if(nil=fieldView) then
  MsgBox.Error("Field View document does not exist", "EPIC-View")
  exit
end
'
'
fieldViewWin=fieldView.GetWin
if(fieldViewWin.IsOpen.Not) then
  fieldViewWin.Open
else
  fieldViewWin.Activate
end
'
'
layersTheme=fieldView.FindTheme(_mainTheme)
'
if(nil=layersTheme) then
  MsgBox.Error("Theme: "+_mainTheme+" does not exist", "EPIC-View")
  exit
end
'
if(layersTheme.IsVisible.Not) then
  layersTheme.SetVisible(True)
end
'
cropTable=layersTheme.GetFTab
selected=cropTable.GetSelection
selected.ClearAll
cropTable.SetSelection(selected)
'
if(layersTheme.IsActive.Not) then
  layersTheme.SetActive(True)
end
'
fieldView.GetDisplay.SetExtent(layersTheme.GetExtent.Scale(1.1))
'av.Run("View.SelectPoint", "")
```

```
' Epic.SoilData
' This Script loads soil and curve number to constant dataset by invoking
' a soil data entry user interface. This menu option is enabled only
' if a user does not haves his own soil files.
' Prepared by Anoop Govil
' Dated 5/18/96
'
command = _exeDir+"soil.exe"
system.execute(command)
_constEnableFlag = 2
_soilEnableFlag = False


' Epic.startUp
' This script creates global variables for various directory paths set by user.
' Prepared by Anoop Govil
' Dated 5/15/96
'
' Global variables used for enabling various menu options.
_selectCellsList = List.Make
_newLineChar = 10.AsChar
_displayEnableFlag = False
_soilDataIsEnabled = False
_constEnableFlag = 0
_soilEnableFlag = False
_runEpicEnableFlag = 0
_opOptionEnableFlag = 0
_userSoilIsAbsent = True
_totalCells = 1
pathsFile = "c:\EVPaths.txt".AsFileName
if(File.Exists(pathsFile).Not) then
  labelList = List.Make
  labelList.Add("Cell Dataset Directory:")
  labelList.Add("Epic Output Directory:")
  labelList.Add("Soil Data Directory:")
  labelList.Add("EPIC Directory:")
  labelList.Add("EXE Directory Name:")
  labelList.Add("Base Dataset Name:")
  labelList.Add("Cell Id Field Name:")
  labelList.Add("Results Table Name:")
  labelList.Add("Main Attribute Table Name:")
  labelList.Add("Main Field View Name:")
  labelList.Add("Main Theme Name:")
  labelList.Add("Main Theme Path:")

  defaultList = List.Make
  defaultList.Add("c:\EPICView\Temp\")
  defaultList.Add("c:\EPICView\Temp\")
  defaultList.Add("c:\EPICView\Soil\")
  defaultList.Add("c:\epic5300\")
  defaultList.Add("c:\EPICView\EXEDir\")
  defaultList.Add("const.dat")
  defaultList.Add("Hru2_")
  defaultList.Add("form.prs")
```

```
defaultList.Add("Attributes of Hru2")
defaultList.Add("Botchlet 1/4 section")
defaultList.Add("Hru2")
defaultList.Add("c:\EPICView\Hru")

userList=MsgBox.MultiInput("Globals Initialization:", "EPIC-View", labelList, defaultList)

if(userList = nil) then
  _cellDatasetDir = defaultList.Get(0)
  _epicOutputDir = defaultList.Get(1)
  _soilDir =  defaultList.Get(2)
  _epicDir = defaultList.Get(3)
  _exeDir = defaultList.Get(4)
  _baseDataset = defaultList.Get(5)

' New additions
  _cellIdFld = defaultList.Get(6)
  _resultsTable = defaultList.Get(7)
  _mainTable = defaultList.Get(8)
  _mainView = defaultList.Get(9)
  _mainTheme = defaultList.Get(10)
  _mainThemePath= defaultList.Get(11)

else
  _cellDatasetDir = userList.Get(0)
  _epicOutputDir = userList.Get(1)
  _soilDir =  userList.Get(2)
  _epicDir = userList.Get(3)
  _exeDir = userList.Get(4)
  _baseDataset = userList.Get(5)

' New additions
  _cellIdFld = userList.Get(6)
  _resultsTable = userList.Get(7)
  _mainTable = userList.Get(8)
  _mainView = userList.Get(9)
  _mainTheme = userList.Get(10)
  _mainThemePath= userList.Get(11)
end
  'write to paths file.
pathFile = TextFile.Make(pathsFile, #FILE_PERM_WRITE)
pathFile.Write(_cellDatasetDir, _cellDatasetDir.Count)
pathFile.WriteElt(_newLineChar)
pathFile.Write(_epicOutputDir, _epicOutputDir.Count)
pathFile.WriteElt(_newLineChar)
pathFile.Write(_soilDir, _soilDir.Count)
pathFile.WriteElt(_newLineChar)
pathFile.Write(_epicDir, _epicDir.Count)
pathFile.WriteElt(_newLineChar)
pathFile.Write(_exeDir, _exeDir.Count)
pathFile.WriteElt(_newLineChar)
pathFile.Write(_baseDataset, _baseDataset.Count)
pathFile.WriteElt(_newLineChar)
pathFile.Write(_cellIdFld, _cellIdFld.Count)
```

```
  pathFile.WriteElt(_newLineChar)
  pathFile.Write(_resultsTable, _resultsTable.Count)
  pathFile.WriteElt(_newLineChar)
  pathFile.Write(_mainTable, _mainTable.Count)
  pathFile.WriteElt(_newLineChar)
  pathFile.Write(_mainView, _mainView.Count)
  pathFile.WriteElt(_newLineChar)
  pathFile.Write(_mainTheme, _mainTheme.Count)
  pathFile.WriteElt(_newLineChar)
  pathFile.Write(_mainThemePath, _mainThemePath.Count)
  pathFile.WriteElt(_newLineChar)
  pathFile.Close
  ' Create a new view at the time of installation.
  fieldView = View.Make
  theSrcName = SrcName.Make( _mainThemePath+" polygon" )
  if (theSrcName = nil) then
    msgbox.Error( "Invalid SrcName", "")
    exit
   end
  mainTheme = Theme.Make( theSrcName )
  mainTheme.SetActive(True)
  mainTheme.SetVisible(True)
  mainThemeLegend = mainTheme.GetLegend
  aField = mainTheme.GetFTab.FindField("Elev")
  mainThemeLegend.Interval(mainTheme.GetFTab, aField, 5)
  mainThemeLegend.SetField(aField)
  mainThemeLegend.RampColors(Color.GetBlue, Color.GetCyan)
  mainTheme.SetName(_mainTheme)
  fieldView.AddTheme(mainTheme)
  fieldView.SetName(_mainView)
  epicProject = av.GetProject
  epicProject.AddDoc(fieldView)
  mainTheme.EditTable
  av.GetProject.Save
else
  eachItem=""
  items=0
  pathFile = TextFile.Make(pathsFile, #FILE_PERM_READ)
  while(pathFile.IsAtEnd.Not)
    aChar = pathFile.ReadElt
    if(aChar = 10.AsChar.AsString) then
      if(items = 0)then
        _cellDatasetDir = eachItem
      elseif(items = 1)then
        _epicOutputDir = eachItem
      elseif(items = 2)then
        _soilDir =  eachItem
      elseif(items = 3)then
        _epicDir = eachItem
      elseif(items = 4)then
        _exeDir = eachItem
      elseif(items = 5)then
        _baseDataset = eachItem
      elseif(items = 6)then
```

```
      _cellIdFld = eachItem
    elseif(items = 7)then
      _resultsTable = eachItem
    elseif(items = 8)then
      _mainTable = eachItem
    elseif(items = 9)then
      _mainView = eachItem
    elseif(items = 10)then
      _mainTheme = eachItem
    elseif(items = 11)then
      _mainThemePath = eachItem
    end
    eachItem=""
    items = items + 1
   else
    eachItem = eachItem + aChar
   end
  end  'end of while loop
end ' End of main if condition
'
' Check if user specified soil is present (used for update property
' of soil data tool option). Also calculate the total number of cells
' present in the gridded coverage.
'
theTable = av.GetProject.FindDoc(_mainTable)
if(nil=theTable)then
  MsgBox.Error("The table: "+_mainTable+", not found.", "Epic")
  exit
end
theTableWin=theTable.GetWin
if (theTableWin.IsOpen.Not)then
  theTableWin.Open
else
  theTableWin.Activate
end
theTableWin.Minimize
'
theVTab = theTable.GetVTab
soilField = theVTab.FindField("Series")
if(soilField <> nil)then
  for each rec in theVTab
    soilSeries = theVTab.ReturnValueString(soilField, rec)
    if(soilSeries.IsNull.Not)then
      _userSoilIsAbsent = False
    end
    _totalCells=_totalCells+1
  end
else
  for each rec in theVTab
    _totalCells=_totalCells+1
  end
end
  theTableWin.Close
```

```
' Epic.updateConstData
' To enable Constant Data Tool menu option.
' Prepared by Anoop Govil
' Dated 5/22/96
'
if (_constEnableFlag = 2)then
  Self.SetEnabled(True)
  exit
elseif (_constEnableFlag = 1 And _userSoilIsAbsent.Not)then
  Self.SetEnabled(True)
  exit
else
  Self.SetEnabled(False)
  exit
end


' Epic.updateGISselection
' To update the menu option Spatial Data.
' Prepared by Anoop Govil
' Dated 2/27/96
'
epicProject=av.GetProject
fieldView=epicProject.FindDoc(_mainView)
'
'
if(nil=fieldView) then
  Self.SetEnabled(False)
  exit
elseif(fieldView.IsActive) then
 layersTheme=fieldView.FindTheme(_mainTheme)
 if(nil=layersTheme) then
   Self.SetEnabled(False)
   Exit
 elseif(layersTheme.IsVisible) then
   Self.SetEnabled(True)
   exit
 else
   Self.SetEnabled(False)
   exit
 end
else
  Self.SetEnabled(False)
  Exit
end
```

```
' Epic.updateRunEpic
' To enable Run Epic menu option.
' Prepared by Anoop Govil
' Dated 5/22/96
'
if(_runEpicEnableFlag = 1) then
  Self.SetEnabled(True)
  exit
else
  Self.SetEnabled(False)
  exit
end


' Epic.updateDispChart
' To update the menu option Chart in Display
' Prepared by Anoop Govil
' Dated 5/22/96
'
epicProject=av.getProject
ResTable=epicProject.FindDoc(_resultsTable)
if(resTable <> nil) then
  loadedResultsTable = True
else
  loadedResultsTable = False
end
fieldView=epicProject.FindDoc(_mainView)
if( (loadedResultsTable) And (_displayEnableFlag) ) then
  Self.SetEnabled(True)
  exit
else
  Self.SetEnabled(False)
  exit
end


' Epic.updateDispMap
' To update the menu option Map in Display
' Prepared by Anoop Govil
' Dated 5/22/96
'
epicProject=av.getProject
fieldView=epicProject.FindDoc(_mainView)
if( (fieldView.GetThemes.Count = 1) And (_displayEnableFlag) ) then
  Self.SetEnabled(True)
  exit
else
  Self.SetEnabled(False)
  exit
end
```

```
' Epic.updateDispTable
' To update the menu option Table in Display
' Prepared by Anoop Govil
' Dated 5/22/96
'
epicProject=av.getProject
ResTable=epicProject.FindDoc(_resultsTable)
if(resTable <> nil) then
  loadedResultsTable = True
else
  loadedResultsTable = False
end
fieldView=epicProject.FindDoc(_mainView)
if( (loadedResultsTable) And (_displayEnableFlag) ) then
  Self.SetEnabled(True)
  exit
else
  Self.SetEnabled(False)
  exit
end



' Epic.updateOpOption
' To enable Output Options menu option.
' Prepared by Anoop Govil
' Dated 5/22/96
'
if(_opOptionEnableFlag = 1) then
  Self.SetEnabled(True)
  exit
else
  Self.SetEnabled(False)
  exit
end


' Epic.updateRemoveThm
' To updates the menu option Remove Themes in Display menu.
' Prepared by Anoop Govil
' Dated 5/22/96
'
epicProject=av.getProject
fieldView=epicProject.FindDoc(_mainView)
if( (fieldView.GetThemes.Count > 1) And (_displayEnableFlag) ) then
  Self.SetEnabled(True)
  exit
else
  Self.SetEnabled(False)
  exit
end
```

```
' Epic.updateSoilData
' To enable Soil Data Tool menu option.
' Prepared by Anoop Govil
' Dated 5/22/96
'
if(_userSoilIsAbsent And _soilEnableFlag)then
  Self.SetEnabled(True)
  _soilIsEnabled=True
  exit
else
  Self.SetEnabled(False)
  exit
end


' Epic.WeatherData
' This Script loads weather file to constant dataset by providing
' a weather data entry user interface.
' Prepared by Anoop Govil
' Dated 5/16/96
'
command = _exeDir+"weather.exe"
system.execute(command)
' Enable Constant Data and Soil Data menu options.
_constEnableFlag = 1
_soilEnableFlag = True
```

```
// Parse.c
/*****************************************************************************
*
*       This program creates a file with comma delemited records which can be loaded back into
*       ArcView as a table.
*
*****************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// Globals!!
char lastChar, CELL_ID_FIELD[20], PRS_FILE[20];
long filePosition=0;
int newLines=0, print=0, firstFiveFields=1, doubleQuotes=0;
int ignoreMoreSpaces=0, numOfCommas=0;
void insertComma(FILE *, char);
int writeToFile(char *, char *, int);
void displayQuit(void);

void main()
{
        int flag=1, retVal;
        FILE *fp, *fpath;
        char path[80], fsel[80], id[10];
        char cellDir[20], epicOutputDir[20], soilDir[20], epicDir[20], exeDir[20], baseData[20];

        strcpy(fsel, "c:\\EVPaths.txt");
        if(!(fpath = fopen(fsel, "r")))
        {
                printf("File %s not found. Aborting...\n", fsel);
                exit(0);
        }
        fscanf(fpath, "%s%s%s%s%s%s%s%s%s", cellDir, epicOutputDir, soilDir, epicDir, exeDir,
                baseData, CELL_ID_FIELD, PRS_FILE);
        strcpy(fsel, cellDir);
        strcat(fsel, "selected.cll");
        if(!(fp = fopen(fsel, "r")))
        {
                printf("File %s not found. Aborting...\n", fsel);
                exit(0);
        }
        fscanf(fp, "%s", path);

        while(!feof(fp))
        {
                fscanf(fp, "%s", id);
                if(strlen(id)==0)break;
                retVal = writeToFile(path, id, flag);
                if(retVal == -1)
                {
                        printf("Error encountered while parsing. Interrupted in middle!\nAborting...\n");
                        displayQuit();
```

```
                        fclose(fp);
                        fclose(fpath);
                        exit(0);
                }
                flag=0;
                strcpy(id, "");
        }
        printf("Successfully completed parsing.\n");
        displayQuit();
        fclose(fp);
        fclose(fpath);
}

/*****************************************************************************************
*       writeToFile()
*
*       This function creates single file with comma delimited records from file(s) created
*       by EPIC as output.
*
*****************************************************************************************/

int writeToFile(char *path, char *cellId, int headerFlag)
{
        int firstTime=1;
        long pos=-2;
        FILE *fileIn, *fileOut;
        char ch=' ', fin[80], fout[80];

        sprintf(fin, "%sform%s.sum", path, cellId);
        sprintf(fout, "%s%s", path, PRS_FILE);
        if(!(fileIn = fopen(fin, "r")))
        {
                printf("File %s not found. Aborting...\n", fin);
                exit(0);
        }

        if(headerFlag)
                fileOut= fopen(fout, "w");
        else
                fileOut= fopen(fout, "a");

        if(headerFlag)
                fprintf(fileOut, "\"%s\",", CELL_ID_FIELD);
        else
        {
                while(fgetc(fileIn) != '\n')
                        if(feof(fileIn))
                        {
                                printf("File %s is empty!\n", fin);
                                return -1;
                        }
                newLines++;
        }
        while(!feof(fileIn))
```

```
	{
		lastChar=ch;
		fscanf(fileIn, "%c", &ch);
		if (firstTime && newLines)
		{
			fprintf(fileOut, "%d,", atoi(cellId));
			firstTime=0;
			ungetc(ch, fileIn);
			ch=' ';
		}
		else
			insertComma(fileOut, ch); // Create a ',' delimited file.
		if(ch == '\n') newLines++;
	}
	fclose(fileIn);
	fclose(fileOut);
	return 1;
}


/*****************************************************************************************
*	insertComma()
*
*	This function processes each character read from the input file(s) and takes action
*	such as inserting ',', ignoring space, writing the character read, etc. depending upon
*	various factors such as	the character read, previous read character, etc.
*
*****************************************************************************************/
void insertComma(FILE *out, char ch)
{
	if(lastChar=='\n')//reset number of commas added.
		numOfCommas=0;
	if( (doubleQuotes) && (ch != ' ') ) //update fileptr if in middle of a quote.
		filePosition--;
	if( (ch=='"') && (doubleQuotes == 1) ) // Inserting ','
	{
		ignoreMoreSpaces=0;
		doubleQuotes=0;
		fprintf(out, "\",");
		numOfCommas++;
		filePosition=0;
	}
	else if( (ch=='"') && (doubleQuotes == 0) ) // Register first ".
	{
			if(lastChar != '"' && lastChar != ' ' && lastChar != '\n')
			{
				ignoreMoreSpaces=0;
				doubleQuotes=1;
				fprintf(out, ",%c", ch);
				filePosition=0;
			}
			else
			{
				ignoreMoreSpaces=0;
```

```
                                doubleQuotes=1;
                                fprintf(out, "%c", ch);
                                filePosition=0;
                        }
        }
        else if( (ch=='\n') && (doubleQuotes == 1) )// Missing second ".
        {
                ignoreMoreSpaces=0;
                doubleQuotes=0;
                filePosition--; //displacement for an extra ',' added.
                fseek(out, filePosition, SEEK_CUR);//wrap back and write \n.
                fprintf(out, "%c", ch);
        }
        else if( (ch=='\n') && (doubleQuotes == 0) && (lastChar == '"') )// To avoid writing ',' after last
                                                                        // field.
        {
                ignoreMoreSpaces=0;
                doubleQuotes=0;
                filePosition--; //displacement for an extra ',' added.
                fseek(out, filePosition, SEEK_CUR);//wrap back and write \n.
                fprintf(out, "%c", ch);
        }
        else if( (ch == ' ') && /*(newLines > 0) &&*/ (!ignoreMoreSpaces) //Insert ',' for numerical fields.
                && (doubleQuotes==0) && (lastChar != '\n') && (lastChar != ' ') && (lastChar != '"'))

        {
                ignoreMoreSpaces=1;
                fprintf(out, ",");
                numOfCommas++;
        }
        else if(lastChar == ' ' && ch == '.') // Add 0 if a float starts with a decimal pt only.
        {
                ignoreMoreSpaces=0;
                fprintf(out, "0%c", ch);
        }
        else if((ch != ' ') && !(ch == '\n' && lastChar == '\n') ) // Ignore all other spaces.
        {
                ignoreMoreSpaces=0;
                fprintf(out, "%c", ch);
        }
}

/*****************************************************************************
*       displayQuit()
*
*       This function displays a message on the screen to prompt the users to close the dos
*       shell window.
*
*****************************************************************************/
void displayQuit(void)
{
        printf("\n\n             ****************************************\n");
        printf("             *                                      *\n");
```

```
        printf("            *   PLEASE CLOSE THIS WINDOW BY    * \n");
        printf("            *                                  *\n");
        printf("            *   CLICKING ON THE EXIT BUTTON.   *\n");
        printf("            *                                  *\n");
        printf("            ************************************\n");
}
```

VITA

Anoop Govil

Candidate for the Degree of

Master of Science

Thesis: EPIC-VIEW: A FULLY INTEGRATED SPATIAL TOOL FOR MODELING SOIL EROSION AND AGRICULTURAL CROP PRODUCTIVITY

Major Field: Computer Science

Biographical:

Personal Data: Born in Sawaimadhopur, Raj., India on December 10, 1969, the son of M.L. Govil and Sudha Govil.

Education: Graduated from St. Xavier's High School, Jamnagar, Gujarat, India; received Bachelor of Engineering degree in Electronics and Communication Engineering from Bangalore University, Bangalore, India in July 1992; completed requirements for the Master of Science degree with a major in Computer Science at Oklahoma State University in July 1996.

Professional Experience: Software Engineer, Quest Infotech (P) Ltd., New Delhi, India, August 1993 to July 1994. Computer Programmer, Agronomy Department, Oklahoma State University, September 1994 to August 1995. Graduate Research Assistant, Computer Science Department , Oklahoma State University, August 1995 to July 1996.