GUI DESIGN AND IMPLEMENTATION

FOR SITES APPLICATION

By

QI LIU

Bachelor of Arts

Suzhou University

Suzhou, Jiangsu, P. R. of China

1993

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

GUI DESIGN AND IMPLEMENTATION

FOR SITES APPLICATION

Thesis Approved:

J Chandler

Thesis Adviser

Michael Nelson

Kathleen M. Kaplan

Thomas C. Collins

Dean of the Graduate College

ACKNOWLEDGMENT

I wish to express my sincere appreciation to my major advisers, Dr. J. P. Chandler and Dr. Mitchell L. Neilsen for their guidance, supervision, encouragement and help for the completion of my thesis work. Their patience and constructive ideas helped me make this thesis work an enjoyable and memorable experience. I consider it a privilege to have worked under their supervision. I would like to express my sincere thanks to Dr. K. Kaplan for serving on my graduate committee. Her support and invaluable suggestion, have helped me to improve the quality of this work. I would like to thank Dr. D. Temple, Plant Science & Water Conservation Laboratory, USDA, for his constructive ideas and support, which proved to be vital during the development stages. I would like to thank Dr. M. L. Neilsen, Dr. J. P. Chandler and the Department of Computer Science for providing me with this research opportunity and their generous financial support.

My greatest appreciation, thanks and love to my wife Jing Xu for all the love, support and inspiration that she has given me. Also thanks to all my friends for their support and much needed help.

TABLE OF CONTENTS

Chapter **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

The Water Resource Site Analysis Computer Program (SITES) was developed to assist the engineer in hydraulic and hydrologic analysis and dams design. The application generates inflow hydrographs. It uses the storage-discharge relationships at dam sites to flood route hydrographs through existing or potential reservoirs. It provides the hydraulic and hydrologic design for Natural Resources Conservation Service (NRCS) dams that have drainage areas ranging from a few acres to over 100 square miles. The application develops the inflow hydrographs from homogeneous subareas, combines them, and routes valley to the dam site.

NRCS distributed the original Structure Site Analysis program (DAMS) to users in 1967. An improved version of the program (DAMS2), released in 1971, incorporated a more flexible input format as well as other improvements to the DAMS program. In 1982 an interim personal computer (PC) version was released. The new version, named SITES, will be released in 1996. This version of SITES accommodates changes in NRCS design criteria and includes the earth spillway erosion technology. Other changes assist with the design of dams that have small drainage areas and sites in complex watersheds, including structures in series.

SITES is based on a FORTRAN program developed and compiled by the US Department of Agriculture (USDA). The previous versions, DAMS and DAMS2 for PC, are both MS DOS-based applications. To design or analyze a job using the program, you need to follow three steps:

- First, you should generate a data file. The data file is a plain text file that can be generated in any text editor. The data file contains all the data the user defines for designing or analyzing specifics and criteria. The file must follow a specific format. Otherwise, the next step cannot be accomplished.
- Second, you should run a program input with the data file you generated in the previous step. The program will output a control file if the input file is correct. The control file contains all the keywords and specific formats that the FORTRAN program requires. All the data that the user defines in step one will be grouped into keyword records. Some graphic files that are used to visualize designing jobs are generated in this step too.
- After you obtain the control and graphic files, you can run the design and analysis program using the control and graphics files as the input file. The design program will output the design and analysis results for users to view and implement the actual project.

The three-step procedure briefly described above has some obvious drawbacks. First of all, this is not a user-friendly approach. A new user may need a quite long learning curve to get used to it. When a user starts learning how to write

the data file, he or she will find that any small error in the data file format or spelling may cause failure in the next step. Even an experienced user cannot guarantee that he will not make any mistakes when creating the data file. Second, when a user uses the program to generate the control file in the second step, if the data file contains any errors, the program will not provide information to explain why the control file cannot be generated. Users may spend a lot of time trying to figure out where the problem lies. Even if the user's data file is all right, he or she may still make mistakes due to misunderstanding of the relationship between the control file and the data file. Some experienced users may directly generate control files manually instead of using the computer program. In that case, the chance for creating errors is much higher, because the control file is much more complicated than the data file. The user must remember how to match data and control keywords and keep in mind many detailed field formats. In the *SITES Water Resource Site Analysis Computer program User's Guide,* more than 200 pages are devoted to the section, *Input Description and Preparation.* It is unreasonable to expect the user to remember so many details or to keep checking those details when designing a project, because this is not what the user should concentrate on. The user's interest is in how to achieve a satisfactory result for a specific dam design and analysis by using a good computer program. The user does not like those restrictions about the input file.

SITES is a powerful application that assists the user in designing and analyzing dams. But the application will have few users if it does not have a user-

friendly tool to generate the input file (control file). Clearly, developing a user-friendly tool to generate the input file or control file is a key issue in updating SITES.

My thesis project designed and implemented a user-friendly control file generating program for SITES. I will discuss the designing and implementing details in the thesis. Specifically, I will describe the major issues involved in the project, discuss previous research and designing reviews, and explain how I solved different kinds of problems.

CHAPTER 2

PROBLEM STATEMENT

As I described earlier, my objective is to develop a user-friendly input file (or control file) generating program for SITES. The SITES Interface Committee has determined that this application will be built and run on the Microsoft Windows operating system. This means that the application is an event-driven computer program. The application will have a graphic user interface (GUI) as the front end. The front end will be used to accept user input data, verify and integrate the data validation. The back end is a set of programming procedures for generating and loading Sites data control files.

The SITES Interface Committee also has determined that, in order to make this application acceptable to the widest range of users, the Graphic User Interface should be able to run on MS Windows 3.x, Windows NT, and Windows 95 operating systems. Microsoft Visual Basic 4.0 for Windows is chosen as the application developing language.

2.1 Control file Definition

The control file consists of a set of records. Every record is headed by a keyword which is used to perform a variety of design and simulation runs. Each control keyword is briefly described below:

SITES	Indicates beginning of job.
STRUCTURE	Loads structure elevation - surface area data table.
WSDATA	Enters design criteria and data for watershed area.
PDIRECT	Enters point design rainfall data.
POOLDATA	Enters principal spillway crest, sediment storage, and valley floor information.
PSDATA	Specifies principal spillway conduit data.
PSINLET	Specifies data for a principal spillway drop inlet riser.
ASCOORD	Defines surfaces of geologic materials in an auxiliary spillway profile by x, y coordinates.
ASMATERIAL	Describes geologic material parameters for each material identified in the ASCOORD table.
ASDATA	Contains additional information for the auxiliary spillway.
ASCREST	Establishes auxiliary spillway crest elevation(s) for the spillway template.
ASSPRFL	Describes the entire auxiliary spillway profile by x, y coordinates.
ASSURFACE	Describe the surface parameters of the auxiliary spillway by x, y coordinates when the spillway location is specified by x, y coordinates.
ASINLET	Provides an inlet profile for the auxiliary spillway channel template.

ASINSURF	Describes surface parameters of the auxiliary spillway inlet channel template.
ASEXIT	Provides parameters characterizing the auxiliary spillway exit channel template.
ASEXSURF	Describes surface parameters of the auxiliary spillway exit channel template.
BTMWIDTH	Establishes auxiliary spillway bottom width(s).
GO,DESIGN	Initiates the design run.
STORM	Allows entry of specific data for a given storm.
RAINTABLE	Enters a specific rainfall distribution.
GO,RAINS	Initiates a run using a series of rainfall amount.
HYD	Enters table of inflow hydrograph coordinates.
GO,STORM	Initiates a run using a selected storm event.
GO,HYD	Initiates a run using an input inflow hydrograph.
GO,TDD	Initiates drawdown computations from a given elevation.
CLPROFILE	Enters coordinates of the embankment centerline profile.
GO,EMB	Initiates embankment quantity computations for given top dam elevations.
ENDTABLE	Signifies end of table.
ENDJOB	Signifies end of job.
ENDRUN	Ends the computer run.

These control keywords are used to identify data records. Every record has a number of fields that represent the user input data. The number of records in a file is determined by specific design or analysis goals. Different situations have different combinations of records (keywords markup the records.) In a specific record, there may be different combinations of fields.

2.2 Current Project Requirement

Because of the complexity of the data sets, this version of the control file generator and GUI will support ten types of SITES runs. Each has been designated as a "Case", i.e., Case A through Case L (Case D and E are repeats of Case B and C.) Every Case represents a unique type of designing run. For example: Case A determines the principal spillway rating and auxiliary spillway crest elevation only; Case B designs auxiliary spillway using spillway templates (ASINLET & ASEXIT) with auxiliary spillway crest elevation known; Case F analyzes an auxiliary spillway for a given storm, etc. The actual design and analysis may not be limited to the ten cases. However, the ten cases are typical design runs.

CHAPTER 3

PREVIOUS RESEARCH AND LITERATURE REVIEW

The previous version, DAMS2, is a text-based application. The application has a character-based user interface which provides limited data editing and data verification functionality. The original character-based user interface is connected to the DAMS2 execution program using an MS DOS batch file. This interface is a starting point for any further improvement or development.

3.1 USDA Team and Other Experts' Research

3.1.1 The work of the SITES Interface Committee

The researchers in the USDA (United States Department of Agriculture) Natural Resources Conservation Service have considered the SITES control file interface issues since 1990. They formed the SITES Interface Committee to direct and promote the building of user-friendly and functionally powerful products. The committee consists of System Analysts, Research Scientists, Hydrologists, Design Engineers, Program Specialists, and Resource Engineers. They have expertise in different aspects of the project, and some of them are also end users of SITES. This provides the development of the SITES control file interface with a very strong technical background.

The SITES Interface Committee has done a lot of research and design work. They redefined the data sets to reflect the new technology adopted in the SITES program. The ten "Case" ("A" though "L") runs are identified as typical runs for SITES. They also developed the text-based control data input screens for the ten types of runs. Although these data input screens do not make an executable program, they can be used as the design prototype. When I began developing the GUI program, these screens served as the basis of the Windows screens.

3.1.2 Experts' Research

Additionally, the SITES Interface Committee is directly involved in the project development process. Some of the members did a lot of in-depth research in dam design, which was incorporated into SITES. Such research also served as the theoretical basis or designing references for SITES and the SITES control file interface project. I will review some of the research.

3.1.2a General descriptions about SITES

To understand the new technology incorporated in SITES, we must mention D.M. Temple, H.H. Richardson, J.A. Brevard, and G.J.Hanson's *SITES: The New DAMS2*(1995). In this article, they explain the three major changes in SITES compared to the old version DAMS.

- Incorporation of vegetal retardance (discharge dependent flow resistance) into a water surface profile routine for use in computing the head discharge rating for the spillway.
- Computation of erosionally effective boundary stress for stability design of the exit channel.
- Evaluation of breach potential using a three-phase erosion modal.

This article also provides an overview of the program changes and their significance to the user.

3.1.2b The headcut technology research

D.H. Temple and G.J. Hanson (1994) also did some in-depth research in the Headcut development in Vegetated Earth Spillways. As they concluded: "For computational purposes, it was found that erosion of vegetated earth spillways could be divided into three phases. These phases are vegetal cover failure, concentrated flow erosion, and headcut advance. A computational procedure is developed for predicting the time associated with minimize data requirements while allowing application to a broad range of conditions. Results of applying the procedure to predict headcut formation are shown to be generally consistent with available field data. This procedure may, therefore, be used to estimate the time of headcut formation for given flow and channel surface conditions." This is an important element for the control file generating project.

Meanwhile, researchers such as D.M. Temple, J.A. Brevard, J.S. Moore, G.J. Hanson, E.H. Grissinger, and J.M. Bradford analyzed vegetated earth spillways. The result of their analysis is the basis of data collection for spillway design and analysis.

3.1.3 The full descriptions of the new SITES technology

The *SITES Water Resource Site Analysis Computer Program User's Guide* version 96.1 is the most important and useful document in designing and implementing the control file generation interface. This User's Guide includes most of the information on the technology used in SITES. In the User's Guide, all control file keywords and related data records and fields are described in text, graphs, and tables. The User's Guide was originally written to help users build control files manually. Although it is very difficult for a user to build a complex data control file manually just by reading the User's Guide, it was a great resource for me when developing the SITES control file generation interface.

3.1.4 Engineer Guide

The division of Water Resources of the Kansas State Board of Agriculture developed the *Engineer Guide 1* and *Engineer Guide 2*. Guide 1 deals with "Earth Dams, Hazard Classes, Spillway Requirements, Detention Storage Requirements, and Rainfall Data." Guide 2 deals with "Administrative Requirements and Criteria for the Design of Earth Dams." These two Guides and the User's Guide helped me understand most of the details about the SITES control data.

3.2 Graphic User Interface and MS Visual Basic

The SITES Interface Committee decided to use Microsoft Visual Basic 4.0 as the developing tool for the control file generation interface project. Visual Basic 4.0 is a powerful Windows application development tool. It allowed me to take great advantages of the Windows application's look and feel. Almost all of the popular and standard Windows application elements, such as controls and dialog boxes, can be developed in a Visual Basic program.

3.2.1 GUI design principles

As Elisabeth Boonin (1986) mentioned, a good GUI application should always keep the user in mind. The user interface is an application's primary mode of communication with the user. Like other forms of communication, whether your ideas will be appreciated depends on how well they are presented. With a good interface design, your program can be efficient and user-friendly. User-oriented is the key element that distinguishes Windows applications from text-based applications.

3.2.1a Apply GUI principles to SITES project

Many experienced GUI developers such as Steve Potts (1996), Michael McKelvy (1996), James A. Dooley (1996), etc. agree that a GUI developer must do a certain level of user analysis and try to balance the needs of different levels of users. When designing the SITES control file interface, I have always remembered and followed this principle. SITES has different levels of users. Some users already have

a lot of experience with the previous versions of DAMS and/or DAMS2. These experienced users may not need many dialog prompts, warning messages, and advanced integrated data checks, which the new and inexperienced users would need.. The SITES committee is working hard to promote the product. They want it to be applied in more and more designing and analysis jobs. This means the products will have more and more inexperienced users in future. To help these inexperienced users, we must build an application with advanced integrated data validation checks along with detailed information presented in the format of both screen dialog boxes and a help system.

According to Elisabeth Boonin (1996), a good user interface should help a user to learn easily and efficiently and allow users to make mistakes. When I worked on this project, I always kept these principles in mind. I also used these principles to measure what I had done.

4.2.1b Frequently used reference books for Visual Basic

Many books have presented very good ideas about developing the graphic user interface. *Microsoft Visual Basic Programmer's Guide, Microsoft Visual Basic Language Reference*, and *Microsoft Visual Basic On-line Documentation*, especially, were important resources for me when developing the SITES control file generation interface.

CHAPTER 4

PROJECT DEVELOPMENT

The SITES control file generation interface project consisted of three parts, or aspects, of programming work. They were Windows GUI design, data management, and control file management. These three parts do not mean that the project involved three separate programs. They are in terms of the project development process and programming model. Throughout the project, the three aspects were integrated together.

I started the project with "Requirements Analysis". In "Requirements Analysis," I obtained the functional requirements for the project from technical experts and documentation in the USDA. After this stage, the project development was a mapping from problem domain to solution domain. Many issues were involved in this process. I will categorize the project development into the three aspects I mentioned above.

4.1 Project Windows GUI design and implementation

Window GUI design was one of the most important parts for the project. I made several major decisions concerning both designing and coding. The following sections explain these decisions.

4.1.1 Combining "Case" screens

As I described above, this application supports nine different "Case" runs. Every "Case" consists of a set of windows screens that allows users to input data for a specific design and analysis. Every screen contains a set of controls that let users input data from the keyboard or choose data from predefined data sets. Every "Case" actually is a path leading to a different set of screens. The first welcome screen contains a menu that let users choose between generating a new control file and opening an existing control file. After the user selects either "New" or "Open" in "File" menu, a screen pops up which allows the user to choose a path. The screen snapshot in Figure 1 shows the screen that contains all the possible selections.

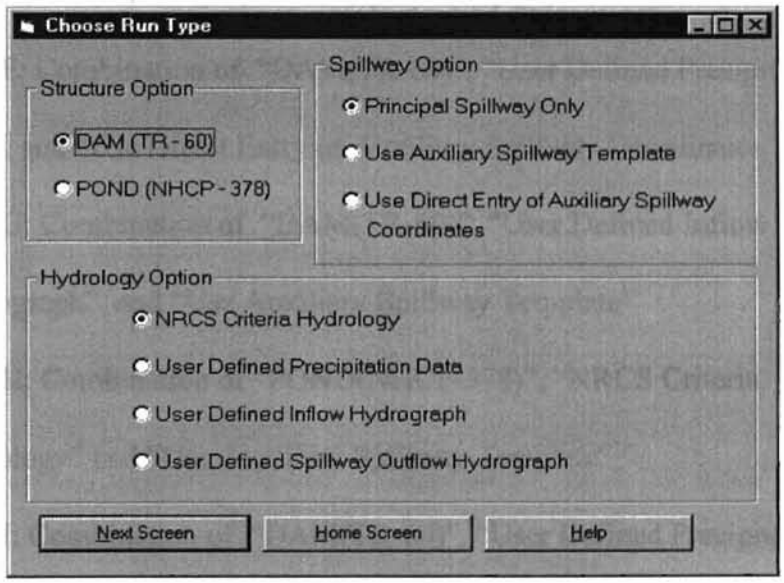


Figure 1. SITES Run Type Screen

Only a subset of possible paths were implemented in this version of the project. If the user chooses a wrong path, a dialog box will pop up to explain the error. The screen design also leaves some space to add "Cases" in the future.

There are three groups of "Radio Button" controls in this screen. They are "Structure Option", "Hydrology Option", and "Spillway Option". The "Radio Button" controls within any one group are mutually exclusive. That means a user can turn on only one "Radio Button" for a group in any particular run. The nine "Case" combinations are listed below:

- Case A: Combination of "DAM(TR-60)" , "NRCS Criteria Hydrology", and "Principal Spillway Only"
- Case B: Combination of "DAM(TR-60)", "NRCS Criteria Hydrology", and "Use Auxiliary Spillway Template"
- Case C: Combination of "DAM(TR-60)", "NRCS Criteria Hydrology", and "Use Direct Entry of Auxiliary Spillway Coordinates"
- Case F: Combination of "DAM(TR-60)", "User Defined Precipitation Data", and "Use Direct Entry of Auxiliary Spillway Coordinates"
- Case G: Combination of "DAM(TR-60)", "User Defined Inflow Hydrograph", and "Use Auxiliary Spillway Template"
- Case H: Combination of "POND(NHCP-378)", "NRCS Criteria Hydrology" and "Use Auxiliary Spillway Template"
- Case I: Combination of "DAM(TR-60)", "User Defined Precipitation Data", and "Use Auxiliary Spillway Template"
- Case J: Combination of "DAM(TR-60)", "User Defined Inflow Hydrograph", and "Use Direct Entry of Auxiliary Spillway Coordinates"

- Case L: Combination of "DAM(TR-60)", "User Defined Spillway Outflow Hydrograph", and "Use Direct Entry of Auxiliary Spillway Coordinates"

When the user chooses a path, all the other screens will follow this specific path. I decided how to deal with three kinds of situations when designing the different screen paths.

4.1.1a Different cases use the same screen

The screen in Figure 2 is an example. All ten case paths will use this same screen.

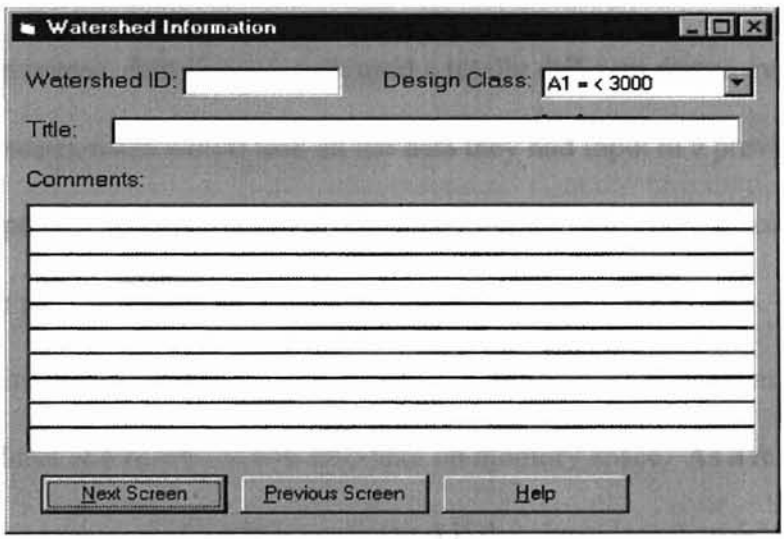


Figure 2. Watershed Information Screen

4.1.1b Different paths use screens that are partially the same

From a programming point of view, building different screens for different paths is easy, but it may cause some serious problems. If I had built different screens for all partially different screens, the application might have had about 200 screens total for all paths, because the control file would be generated after the user goes through the entire specific path. The client, the SITES interface committee, required that the user should be allowed to change paths even when he or she is in the middle of a specific path, and the user should be able to keep the previously input data for use in another path if the specific data field is the same. To meet this requirement, I included the "Previous Screen" button in every screen to allow the user to go back to the path selection screen. But if every path used a totally different screen even for partially same screens, users would lose all the data they had input in a previous path.

Another problem is that if a user keeps changing the paths before going through an entire path, he or she may load all 200 screens into the memory. The average screen size is 20KB. So 200 screens would require at least 4MB memory, and other procedures and functions will also take up memory space. As a result, users might have had a problem running this application if they have less than 6 MB memory.

I finally decided to adopt another approach. I designed the same screen for different paths if these paths share part of the data in a screen. This design makes the screen dynamically changeable at run time. The following is an example of such screens. Both "Case A" and "Case B" have "Rainfall Data" screen, but "Case B" has

more data field than "Case A." I used one screen for the two cases. If the user reaches the screen from "Case A", only one screen tab is visible, and it contains the data fields for "Case A" only.

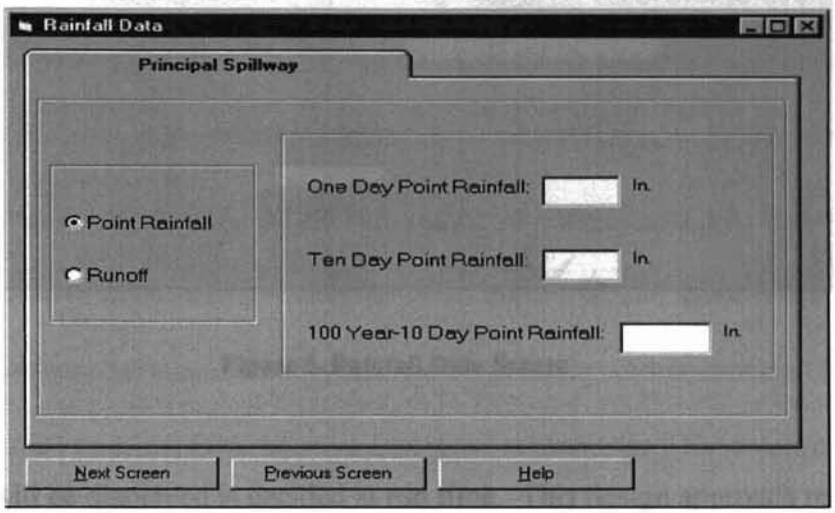


Figure 3. Rainfall Data Screen

If the user chooses "Case B", the same screen as "Case A" will display two tab folders:

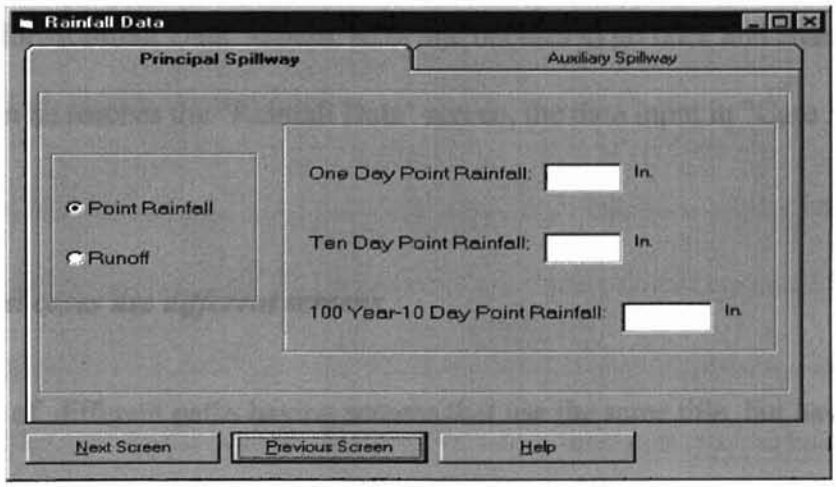


Figure 4. Rainfall Data Screen

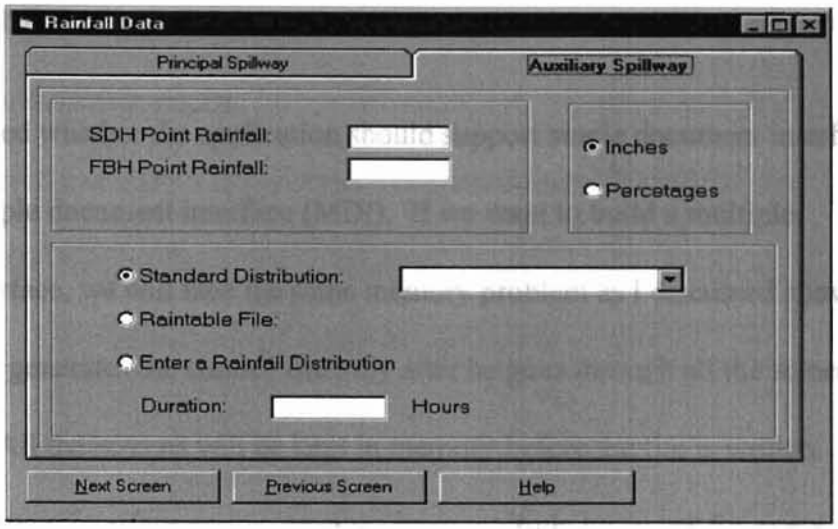


Figure 5. Rainfall Data Screen

What will be displayed is decided at run time. This design approach reduced the total number of screens to about forty, which consumes much less system resources. With this design, the user is also able to use previously input data should he or she change path.

In the previous example, if the user chooses "Case A" at first, but after inserting data for "Rainfall Data" screen, he or she decides to go back and change to "Case B", when he reaches the "Rainfall Data" screen, the data input in "Case A" will still be there.

4.1.1c Different cases use different screens

In case of different paths having screens that use the same title, but having different data fields, I decided to design different screens. In this case, the user would not expect to use any data field for a different path if no data fields are the same.

4.1.2 SDI vs. MDI

I decided whether the application should support single document interface (SDI) or multiple document interface (MDI). If we want to build a multiple document interface, we will face the same memory problem as I discussed above, because a user generates the control file only after he goes through all the screens along a path. All the screens will be kept in memory before the file is written. Multiple document interface means we must have multiple instances for every visual basic form and other controls. This may very easily cause memory shortage. The MDI approach also makes it difficult to maintain the reusability of the data from different paths.

On the other hand, the single document interface approach uses less memory and simplifies path maintenance. Because users do not have much need for generating multiple files at the same time, I decided to build a single document interface for this project.

4.1.3 One screen, multiple functionality

I strived to maintain multiple screen appearances at run time. Many screens were implemented to interact with the user at run time. Screens will automatically change their appearances according to the user's requirements or selection. This approach provided one screen with the functionality of several screens and minimized the number of screens for each path. A typical example is the "Topsoil Fill and General Fill" screen. This screen has multiple functions. It could be implemented as

several screens, but all the functions were implemented in one screen. Below are several snapshots for this screen:

The screen in Figure 6 appears when the user selects "None" for both "Topsoil Fill" and "General Fill."

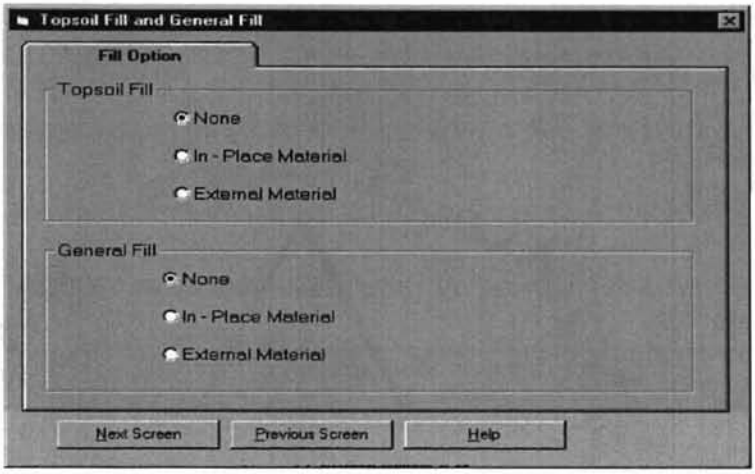


Figure 6. Topsoil Fill and General Fill Screen Appearance 1

The screen in Figure 7 appears when the user chooses "In-Place Material" from "Topsoil Fill."

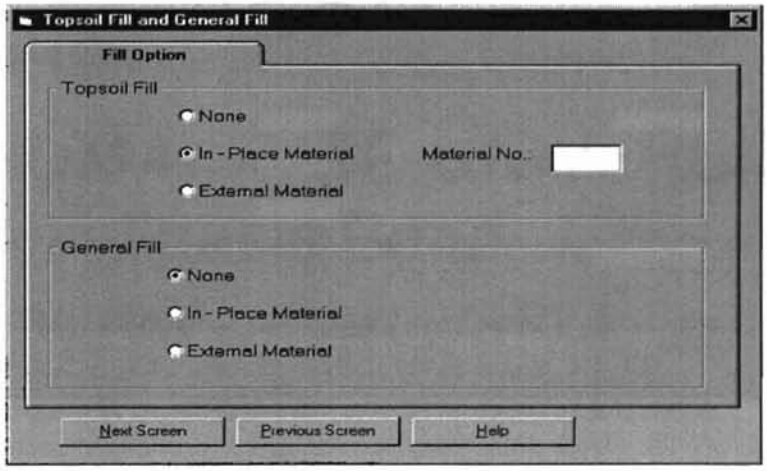


Figure 7. Topsoil Fill and General Fill Screen Appearance 2

The two screens in Figure 8 and Figure 9 are for the user to choose "External Material" from "Topsoil Fill."

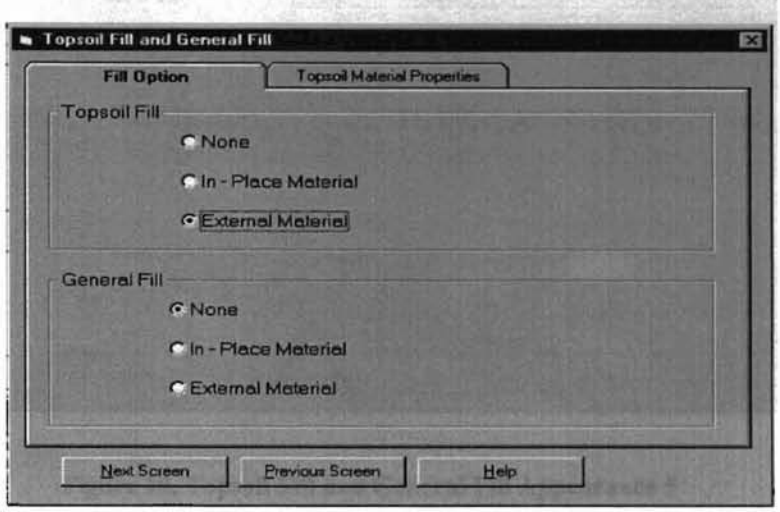


Figure 8. Topsoil Fill and General Fill Screen Appearance 3

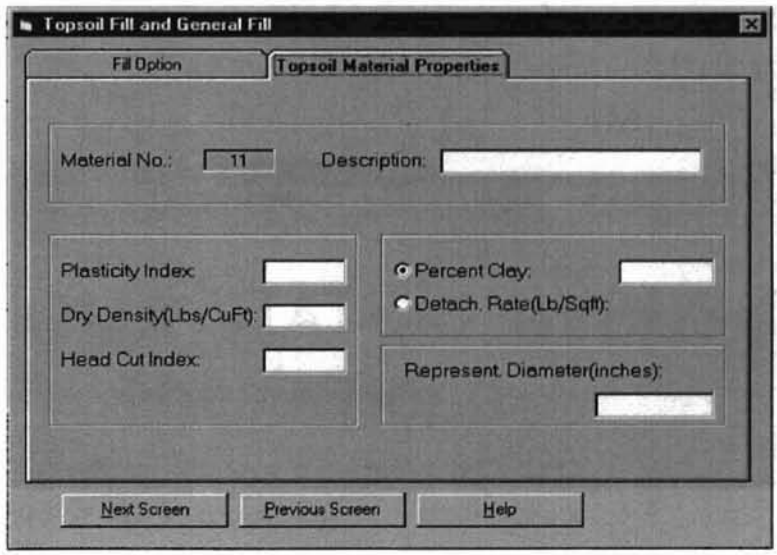


Figure 9. Topsoil Fill and General Fill Appearance 4

The screen in Figure 10 is displayed when the user chooses "In-Place" from "General Fill."

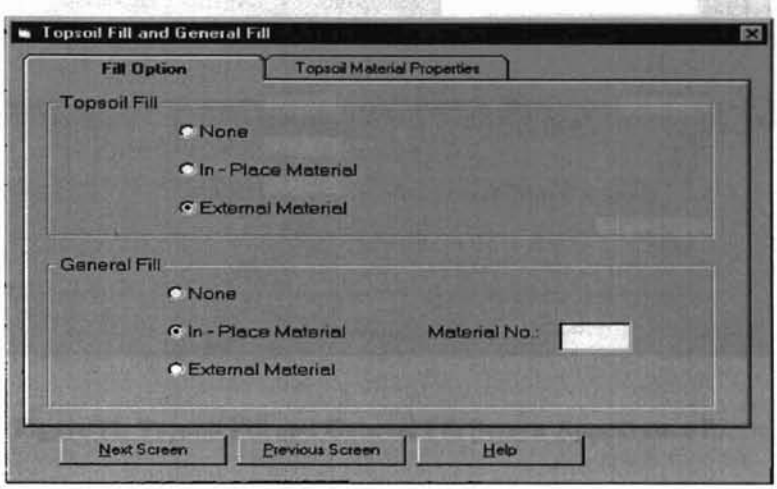


Figure 10. Topsoil Fill and General Fill Appearance 5

The following two screens allow the user to choose "External Material" from "General Fill."

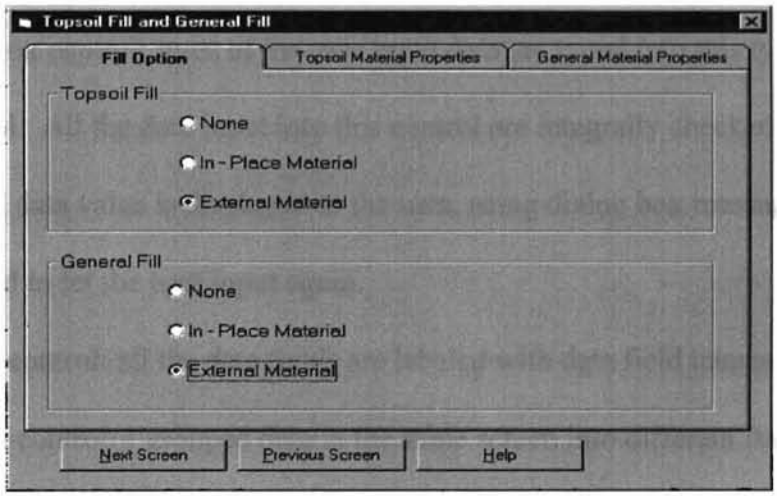


Figure 11. Topsoil Fill and General Fill Screen Appearance 6

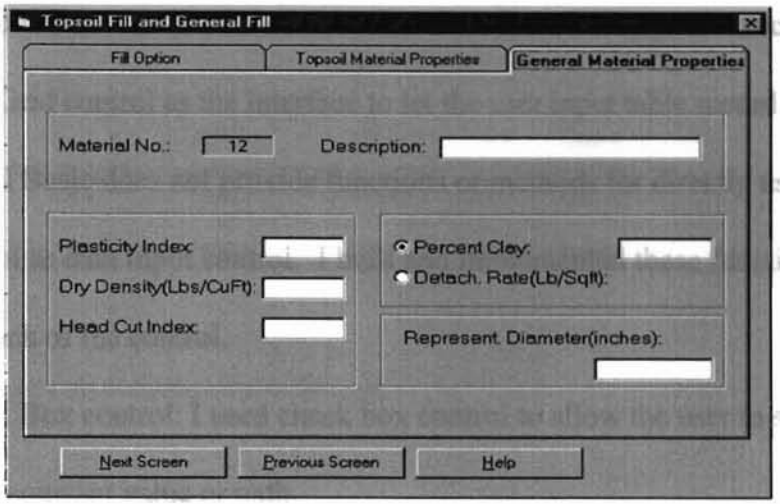


Figure 12. Topsoil Fill and General Fill Screen Appearance 7

4.1.4 Use Visual Basic Controls

Microsoft Visual Basic Version 4.0 provides many windows controls for building Windows applications. I used twelve different types of controls in this project. Some of the major controls are listed below.

- TextBox control: most of the user input data are typed into this type of control. All the data input into this control are integrally checked. Any illegal data value is displayed to the user, using dialog box messages, and cleared to let the user input again.
- Label control: all the data fields are labeled with data field names.
- Frame control: I grouped data in the same screen into different frames according to data logic category or control type.
- ComboBox control: When data fields have predefined values, this control lets the user select data instead of directly inputting data.

- Grid control: The SITES control data file contains many table records. I used Grid control as the interface to let the user input table record data. Visual Basic does not provide functions or methods for directly using grid control as data input control. I built and implemented these functions at the back of the control.
- Check Box control: I used check box control to allow the user to select some constant value or path.
- OptionButton control: This control has the same function as check box control, but it was used for mutually exclusive values or paths.
- SSTab control: This control divides data into different tab pages. I used this control to make a screen contain more information.

4.1.5 Extend Visual Basic "Grid" Control Functionality

Visual Basic comes with a rich set of controls. Some of these controls have good appearance but limited functionality that may not be suitable for some specific programming requirements. The "Grid" is one of this kind of controls.

Many data records in Sites control file are "Table" records. "STRUCTURE", "RAINTABLE", "ESCOORD", and "ESMATERIAL" are typical table records. For example, the "STRUCTURE" record may appear in control file like Figure 13.

```
STRUCTURE  154B4    Site Data From TR48 1971 Version
                    591        76.6      48.2      247.8
                    593        106.9     67.8      304.5
```

595	142.8	69.3	320.4
601	262.7	71.2	322.9
605	360.3	78.6	335.6
609	456.9	80.2	347.3
615	607.7	73.2	385.3

ENDTABLE

Figure 13. Structure Table Record

In this table, there can be up to five fields and twenty data records. The five data fields are "Elevation (feet)", "Surface Area (Acres)", "PS Discharge (CFS)", "Storage Vol.(Acre Feet)", and "AS Discharge (CFS)". Obviously, providing a table-like interface is an effective way for users to input data.

Visual Basic provides two kinds of table controls. One is "Grid" control , another is "Data-Bound Grid" control. The "Data-Bound Grid" control is designed for loading Database table. The "Grid" control is for displaying data from any kind of resources. I used "Grid" control for most of the "Table" input interface. Following is a snapshot of the "STRUCTURE" table input screen:

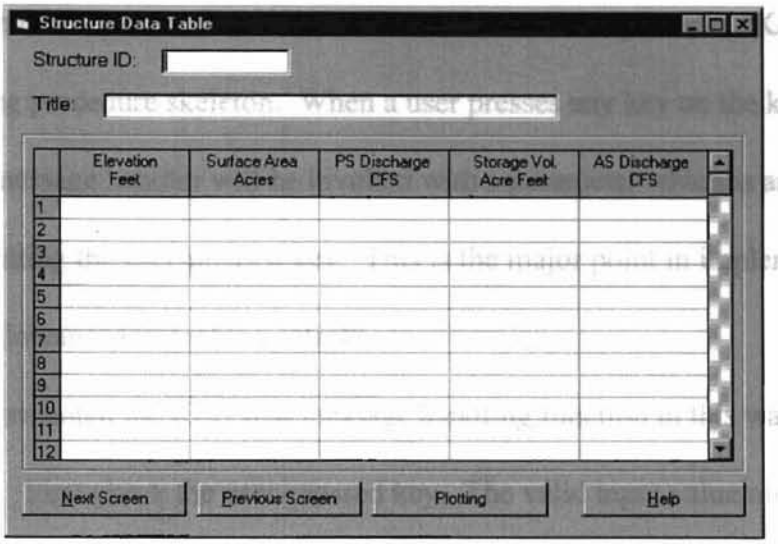


Figure 14. Structure Table Input Screen

One of the weaknesses of Visual Basic "Grid" control is that it does not provide any function for a user to input data directly into a table. That means a user cannot type any character into a table cell. The major function of "Grid" control is for displaying data instead of inputting data. In this application, I need to provide a table interface to let the user input and reload data. Using "Grid" control to reload data does not cause any problem. But how to input data? This is a problem. There are several choices for solving the problem. One way is to buy a third party "OCX" control that has the functionality I need. Or I could have built a new custom control using other tools if time and money had allowed me to do so. Another way is to extend the functionality of "Grid" control to make it work as I want. I chose the third solution. In this application, I successfully extended "Grid" control to have "Input" functionality. I will explain briefly how I extended "Grid" control.

Like most Windows controls, the "Grid" control also provides "KeyPress" event handling procedure skeleton. When a user presses any key on the keyboard, a "KeyPress" message handler will be invoked with a parameter which is an ASCII value representing the user-pressed key. This is the major point in implementing a "user input allowance Grid table control".

I implemented the KeyPress message handling function in this way:

- First check the user-pressed key. The valid input value must be numeric character ("0" to "9"), point ("."), "Enter" key, or "Backspace" key. Any other key pressed by the user will not be displayed in the table cell.
- If the user presses an "Enter" key in keyboard, the data in the current cell will be checked according to some specific integrated data rules. If the data are valid, the next cell will get focus to let the user input data. Otherwise the data in the current cell will be cleared and the focus will still be set at the current cell to let the user input data again.
- If the user presses a key that is any one of "0" to "9" or decimal point ".", the character will be added to a temporary string variable; then the string variable will be displayed in the current table cell.
- Whenever the table changes focus from one cell to another, the temporary string value will be changed to hold new cell data (string).
- If the user pressed the "Backspace" key, the temporary string which holds the current cell data will be cut till the last character and

displayed in the cell. This gives the user chances to remove or change the data in any cell.

Figure 15 is the list of "Key Press" event handler function source code for the "STRUCTURE" table:

```
Private Sub grdTable_KeyPress(KeyAscii As Integer)
'Accept user input for all the fields in table
Dim NextCol As Integer
Dim NextRow As Integer

Enter = False
 'In case the user pressed "Enter" key, the data will be integrally
 'checked.  If data is valid.  The table cell will change to next.
 If KeyAscii = 13 Then
 Enter = True
grdTable.Row = grdTable.Row - 1
If grdTable.Row <> 0 Then
If Val(grdTable.Text) >= Val(Temp) Then
MsgBox "The value must be increased from previous; try again", 48, "Valid data"
grdTable.Row = grdTable.Row + 1
 grdTable.Text = ""
 Temp = ""
Else
grdTable.Row = grdTable.Row + 1
NextCol = grdTable.Col + 1
NextRow = grdTable.Row + 1
If NextCol >= grdTable.Cols And NextRow < grdTable.Rows Then
 grdTable.Col = 1
grdTable.Row = NextRow
```

```
 grdTable.SelStartCol = grdTable.Col
grdTable.SelStartRow = grdTable.Row
grdTable.SelEndCol = grdTable.Col
grdTable.SelEndRow = grdTable.Row
ElseIf NextCol < grdTable.Cols Then
grdTable.Col = NextCol
grdTable.SelStartCol = grdTable.Col
ElseIf NextCol >= grdTable.Cols And NextRow >= grdTable.Rows Then
 grdTable.Col = 1
 grdTable.Row = 1
grdTable.SelStartCol = grdTable.Col
grdTable.SelStartRow = grdTable.Row
grdTable.SelEndCol = grdTable.Col
grdTable.SelEndRow = grdTable.Row
End If
End If

Else
grdTable.Row = grdTable.Row + 1
NextCol = grdTable.Col + 1
NextRow = grdTable.Row + 1
If NextCol >= grdTable.Cols And NextRow < grdTable.Rows Then
grdTable.Col = 1
grdTable.Row = NextRow
grdTable.SelStartCol = grdTable.Col
grdTable.SelStartRow = grdTable.Row
grdTable.SelEndCol = grdTable.Col
grdTable.SelEndRow = grdTable.Row
ElseIf NextCol < grdTable.Cols Then
grdTable.Col = NextCol
```

```
grdTable.SelStartCol = grdTable.Col
grdTable.SelStartRow = grdTable.Row
ElseIf NextCol >= grdTable.Cols And NextRow >= grdTable.Rows Then
grdTable.Col = 1
grdTable.Row = 1
grdTable.SelStartCol = grdTable.Col
grdTable.SelStartRow = grdTable.Row
grdTable.SelEndCol = grdTable.Col
grdTable.SelEndRow = grdTable.Row
End If
End If

'In case the user pressed the backspace key, the current cell data will
'be deleted one character a time.
ElseIf KeyAscii = 8 Then
If grdTable.Text <> "" Then
Temp = grdTable.Text
End If
If Len(Temp) <> 0 Then
Temp = Left(Temp, Len(Temp) - 1)
grdTable.Text = Temp
End If

'Any key press that represents ASCII code less than 48 and not
'equal to 46 is not valid data.
ElseIf KeyAscii <> 46 And KeyAscii < 48 Then
KeyAscii = 0
Beep
'Any key press that represents ASCII code larger than 57
'is not valid data.
```

```
ElseIf KeyAscii > 57 Then
KeyAscii = 0
Beep
'Table cell will show up valid data input
Else
Temp = Temp + Chr(KeyAscii)
grdTable.Text = Temp
End If
End Sub
```

Figure 15. Key Press Event Handler for Structure Table

Besides the "Key Press" event handling, I also implemented the "Grid" control "GotFocus" event and "SelChange" event handling functions. The "GotFocus" event handling is for setting focus on the current cell in the "Grid" control. The "SelChange" event handling is for clearing the temporary string that holds the user input data. The cleared string will be used to hold the user input for a new cell. Figure 16 and Figure 17 are the source code for these two event handlings:

```
Private Sub grdTable_GotFocus()
'when grdtable get focus, all cells need to be
'set to the current cell of the table.
grdTable.SelEndCol = grdTable.Col
grdTable.SelStartCol = grdTable.Col
grdTable.SelStartRow = grdTable.Row
grdTable.SelEndRow = grdTable.Row
End Sub
```

Figure 16. Structure Table Get Focus Event Handler

```
Private Sub grdTable_SelChange()
'Temp hold the string user input into the table cell.
'This value must be clear when cell is changed.
 Temp = ""
End Sub
```

Figure 17. Structure Table sel Change Event Handler

Thus, I created a "Grid" control that can be used not only to display data in a table, but also to let a user input data directly into a table cell. After the extension, the "Grid" control becomes an interactive interface between the user and the computer program.

The "STRUCTURE" is a typical "Table" interface in the project. Many other data records in SITES's control file need table-like interface for a user to input data. They are implemented in a way similar to the "STRUCTURE" table.

4.2 Data Validation and Management

Data are the core of the application. One of the goals for this application is to help users generate correct control files. A correct control file is based on the correct data set inputted by users. So the data validation check is a very important task for this application. Actually, about one third of the program source code involved data validation checking. Data validation is also one of the hardest part of the project to program.

This application has three levels of data validation. They are data field level, screen level, and application level. The following sections explain these three levels of data validation.

4.2.1 Field level data validation

According to *SITES Water Resource Site Analysis Computer program User's Guide* , most of the data fields in this application have some kind of constraint. Those constraints are the basis of field level data validation. Beside the data constraints, all the data fields have some kind of valid range requirement, such as string data length, and minimum and maximum values for numeric data, etc.

4.2.1a Data field constraint enforcement

All the user-input data must appear on the screen as visible characters. The *Guide* defines seven conventions which represent seven types of data characters:

- A = Alphanumeric (letters or numeric) data that may appear in any location within specified columns.
- N = Numeric data that may appear in any location within specified columns and may include special characters such as decimal points and plus or minus signs. Commas are not allowed.
- I = Integer numbers (no decimal points) that may occur in any location within specified columns.

- Ix = Integer numbers (no decimal points) that must occur in a fixed right justified position.
- e = Essential data for use of control word. Program prints error massages and normally terminates job execution if user omits the data.
- - = Negative, negative data required.
- (-) = Negative, negative data allowed.

I enforced these data constraint for the data in SITES control files that have those requirements. In most cases, I used the "KeyPress" event handling function as filters to enforce the data constraints. Any user-pressed key which is not a valid character for the specific data will be filtered out in a "KeyPress" event handling function. The following is an example.

There is a "Number of Conduit" field in The "PSDATA" record. This field requires the data value to be either an "integer" or a "real". This means that only "0"~"9" and "." can be the valid input character. Any key press that is not related to the valid data, "Enter" key press, or "Backspace" key press will be filtered out.

The source code listed in Figure 18 shows the implementation of the data filter idea. Most of the data inputting in this project used the same technique.

```
Private Sub txtNumOfConduit_KeyPress(KeyAscii As Integer)
'If a user pressed "Enter" key (ASCII code 13), the focus will
'be changed to another control
If KeyAscii = 13 Then
KeyAscii = 0
Enter = True
 txtNumOfConduit_LostFocus
```

```
'Any Key Press that is less than ASCII 48 and not equal
'to ASCII code 8 (Backspace key) or 46 (".") will be filtered out
ElseIf KeyAscii < 48 And KeyAscii <> 8 And KeyAscii <> 46 Then
KeyAscii = 0
Beep
'Any Key Press that is larger than ASCII code 57 will be filtered out
ElseIf KeyAscii > 57 Then
KeyAscii = 0
Beep
End If
End Sub
```

Figure 18. Example Of Data Filter In Key Press Event Handler

4.2.1b Input data validation check

Besides the enforced data type from inputting, I also implemented data validation checks in this application. Almost all the data fields in this application have some predefined data range requirement such as the maximum and minimum values for numeric data, maximum character string length for string data, etc.

The data field validation check works in this way: whenever a user finishes inputting a data value for a field and hits the "Enter" key on keyboard, the "LostFocus" event handler is invoked to handle the situation. One of the tasks for the "LostFocus" event handler is to check if the data which the user input is in the valid range or not. If the data checked is not in the valid range, the data in current control will be cleared out and focus will still be set on the current control. Otherwise the focus will be set on another data field.

I will still use the "txtNumOfConduit" control field as an example. In the previous code listing, I already demonstrated that when the user hits the "Enter" key, the *txtNumOfConduit_LostFocus* handler is invoked. The following is the code for this event handling function.

```
Private Sub txtNumOfConduit_LostFocus()
'Check whether the user input value is valid or not.  If the
'Value is valid, set focus to txtLengthOfConduit control.
'Otherwise clear the current control content and prompt
'User with a message box.  Focus will still be set on current
'Control

Dim value                    'hold the input value
Dim OutRange As Boolean   'Value is out of range or not

OutRange = False
If txtNumOfConduit.Text <> "" Then
value = Val(txtNumOfConduit.Text)
If value < 1 Then
MsgBox "Minimun Number Of Conduit is 1, try again.", 48, "Number of Conduit"
OutRange = True
txtNumOfConduit.Text = ""
End If
End If

If Enter = True And OutRange = False Then
Enter = False
txtLengthOfConduit.SetFocus
End If
```

```
End Sub
```

Figure 19. Example Of Data Filter In Lost Focus Event Handler

This is a simple case for field level data validation check. There were some cases that required much more complicated data checks, but the principles are just the same.

4.2.2 Screen Level Data Validation

Because the field level data validation check depends on the "Enter" key press generating "LostFocus" event handler, we can't guarantee the input data will always be checked at field level. If a user uses a "mouse" click instead of an "Enter" key press to change focus from one control to another, the Data field might not be checked properly. Some data fields are required data for a specific record, which means the data field should not be left blank, if a user uses a mouse click to move focus from one field to another and finally clicks "Next Screen" button to jump to another screen, the required data field might be left blank forever. To handle such cases, I implemented the screen level data check.

It seems the screen level data check is just a repeat of the field level data check. Doing this is to guarantee that all the data fields will be checked before the current screen progresses to another screen. This is true for some of the screens. But for any screen that contains the "Grid" control, this is not true.

As I described above, I extended the "Grid" control to use it as the user input data container. I implemented some field level data checks when a user presses the

"Enter" key to change from the current cell to the next cell in the "Grid" table. There is also the possibility for some cells never to be checked if a user uses a mouse click or "arrow" key to change focus from one cell to another. The screen level data check is needed to solve this problem, as I already mentioned.

In addition, we have a new problem for "Grid" table internal data check. If a table has multiple columns and rows, we have to maintain many inter-relationships among table cells. This make things much more complicated than a single field data validation check. The best place to accomplish this is in screen level instead of the cell level. In a screen level check, the user has already finished the table data input and is ready to go to the next screen. Because the user does not plan to input any data into the table at this point, it is the right time to do the integrated data check for the table cells containing data.

Let me continue to use the "STRUCTURE" table as an example. In this table, all the column value must be in increase order when entered. The "Elevation" , "Surface Area" or "Storage Vol." fields require at least two entries. The "Surface Area" and "Storage Vol." are mutual exclusion fields. Only one of the fields should appear in a table. "PS Discharge" and "AS Discharge" are optional fields, but once entries are begun, they should continue to the end of the table. The last value of "Elevation" and "Surface Area" or "Storage Vol." is the mark of end of table. All these data requirements are enforced in a screen level data check. I implemented the screen level data check in the "Next Screen" button click event handling procedure. Figure 20 is the source code of the procedure:

```
Private Sub cmdNext_Click()
'For Case A B C F G H I J
Dim counter As Integer
'values1 and value2 are  temporary value holders
Dim value1 As Double
Dim value2 As Double
'hole is for deciding if table column value is continued or not
Dim hole As Boolean
'holeCounter is for counting number of holes between two data
Dim holeCounter As Integer
'ComString is for hold cases string
Dim ComString As String
'The four boolean values judge if the columns are input
Dim surface As Boolean
Dim storage As Boolean
Dim discharge As Boolean
Dim ascharge As Boolean
'Count the length of table rows that contain data
Dim length As Integer

DAMS2.bypass = False   'Global value for special case flag
surface = False
storage = False
discharge = False
ascharge = False
AScharging = False
holeCounter = 0
length = 0
ComString = "ABCFI"
hole = False
```

```
'check data validation before going to the next screen
grdTable.Col = 1
grdTable.Row = 1
If grdTable.Text = "" Then
 MsgBox "First entry is required for this column", 48, "Input Data"
GoTo dataReIn
Else
value1 = Val(grdTable.Text)
minElev = value1
maxElev = value1
End If
length = length + 1
grdTable.Row = 2
If grdTable.Text = "" Then
MsgBox "At least two entries requried for this column", 48, "Input Data"
GoTo dataReIn
Else
value2 = Val(grdTable.Text)
If value1 >= value2 Then
MsgBox "The value must be increased from previous row", 48, "Data check"
grdTable.Text = ""
Temp = ""
GoTo dataReIn
Else
value1 = value2
maxElev = value2
End If
End If
length = length + 1
For counter = 3 To 20 Step 1
```

```
grdTable.Row = counter
If grdTable.Text = "" Then
hole = True
holeCounter = holeCounter + 1
Else
If hole = True Then
grdTable.Row = grdTable.Row - holeCounter
MsgBox "This cell must contains data", 48, "Data Check Message"
GoTo dataReIn
End If
value2 = Val(grdTable.Text)
If value1 >= value2 Then
MsgBox "The value must be increased from previous row", 48, "Data check"
grdTable.Text = ""
 Temp = ""
GoTo dataReIn
Else
value1 = value2
maxElev = value2
End If
length = length + 1
End If
Next

grdTable.Col = 2
For counter = 1 To 20 Step 1
grdTable.Row = counter
If grdTable.Text <> "" Then
surface = True
End If
```

```
Next
grdTable.Col = 3
For counter = 1 To 20 Step 1
grdTable.Row = counter
If grdTable.Text <> "" Then
discharge = True
End If
Next

grdTable.Col = 4
For counter = 1 To 20 Step 1
grdTable.Row = counter
If grdTable.Text <> "" Then
storage = True
End If
Next

If grdTable.Cols = 6 Then
grdTable.Col = 5
For counter = 1 To 20 Step 1
grdTable.Row = counter
If grdTable.Text <> "" Then
ascharge = True
AScharging = True
End If
Next
End If

'if PS Discharge column has data, several screens will be bypassed
If discharge = True Then
```

```
DAMS2.bypass = True
End If

'work on case of user inputting both surface and storage data.
'user must make decision to choose either one from the two
If surface = True And storage = True Then
frmTableDialog.Show 1
If frmTableDialog.Tag = "surface" Then
grdTable.Col = 4
For counter = 1 To 20 Step 1
grdTable.Row = counter
grdTable.SelStartCol = grdTable.Col
grdTable.SelStartRow = grdTable.Row
grdTable.SelEndCol = grdTable.Col
grdTable.SelEndRow = grdTable.Row
grdTable.Text = ""
Next
storage = False
Else
grdTable.Col = 2
For counter = 1 To 20 Step 1
grdTable.Row = counter
grdTable.SelStartCol = grdTable.Col
grdTable.SelStartRow = grdTable.Row
grdTable.SelEndCol = grdTable.Col
grdTable.SelEndRow = grdTable.Row
grdTable.Text = ""
Next
surface = False
End If
```

```
End If

'work on case of user's choice of surface data
hole = False
holeCounter = 0
If surface = True And storage = False Then
grdTable.Col = 2
grdTable.Row = 1
If grdTable.Text = "" Then
MsgBox "First entry is required for this column", 48, "Input Data"
GoTo dataReIn
Else
value1 = Val(grdTable.Text)
End If
grdTable.Row = 2
If grdTable.Text = "" Then
MsgBox "This cell must contains data", 48, "Data Check Message"
GoTo dataReIn
Else
value2 = Val(grdTable.Text)
If value1 >= value2 Then
MsgBox "The value must be increased from previous row", 48, "Data check"
grdTable.Text = ""
Temp = ""
GoTo dataReIn
Else
value1 = value2
End If
End If
```

```
holeCounter = 0
For counter = 3 To length Step 1
grdTable.Row = counter
If grdTable.Text = "" Then
hole = True
holeCounter = holeCounter + 1
Else
If hole = True Then
grdTable.Row = grdTable.Row - holeCounter
MsgBox "This cell must contain data", 48, "Data Check Message"
GoTo dataReIn
End If
value2 = Val(grdTable.Text)
If value1 >= value2 Then
MsgBox "The value must be increased from previous row", 48, "Data check"
grdTable.Text = ""
Temp = ""
GoTo dataReIn
Else
value1 = value2
End If
End If
Next
If length + 1 <= 20 Then
grdTable.Row = length + 1
End If
If grdTable.Text <> "" Then
MsgBox "The number of Surface Area data must be less than or equal to that of Elevation
data",    48,  "Data check"
End If
```

```
For counter = length + 1 To 20 Step 1
grdTable.Row = counter
grdTable.SelStartCol = grdTable.Col
grdTable.SelStartRow = grdTable.Row
grdTable.SelEndCol = grdTable.Col
grdTable.SelEndRow = grdTable.Row
grdTable.Text = ""
Next
End If

'work on case of user's choice of storage data
hole = False
holeCounter = 0
If surface = False And storage = True Then
 grdTable.Col = 4
grdTable.Row = 1
If grdTable.Text = "" Then
MsgBox "First entry is required for this column", 48, "Input Data"
GoTo dataReIn
Else
value1 = Val(grdTable.Text)
End If
grdTable.Row = 2
If grdTable.Text = "" Then
MsgBox "This cell must contain data", 48, "Data Check Message"
GoTo dataReIn
Else
value2 = Val(grdTable.Text)
If value1 >= value2 Then
MsgBox "The value must be increased from previous row", 48, "Data check"
```

```
grdTable.Text = ""
Temp = ""
GoTo dataReIn
Else
value1 = value2
End If
End If

holeCounter = 0
For counter = 3 To 20 Step 1
 grdTable.Row = counter
If grdTable.Text = "" Then
hole = True
holeCounter = holeCounter + 1
Else
If hole = True Then
 grdTable.Row = grdTable.Row - holeCounter
MsgBox "This cell must contain data", 48, "Data Check Message"
GoTo dataReIn
End If
value2 = Val(grdTable.Text)
If value1 >= value2 Then
MsgBox "The value must be increased from previous row", 48, "Data check"
grdTable.Text = ""
Temp = ""
GoTo dataReIn
Else
value1 = value2
End If
```

```
End If
Next
If length + 1 <= 20 Then
grdTable.Row = length + 1
End If
If grdTable.Text <> "" Then
MsgBox "The number of Storage Area data must less than or equal to that of Elevation data",
48, "Data check"
End If
For counter = length + 1 To 20 Step 1
grdTable.Row = counter
grdTable.SelStartCol = grdTable.Col
grdTable.SelStartRow = grdTable.Row
grdTable.SelEndCol = grdTable.Col
grdTable.SelEndRow = grdTable.Row
grdTable.Text = ""
Next
End If

'work on case of PS Discharge data is present
hole = False
holeCounter = 0
If discharge = True Then
grdTable.Col = 3
grdTable.Row = 1
   value1 = Val(grdTable.Text)
   grdTable.Row = 2
   If grdTable.Text = "" Then
     MsgBox "This cell must contain data", 48, "Data Check Message"
     GoTo dataReIn
```

```
Else
 value2 = Val(grdTable.Text)
 If value1 >= value2 Then
  MsgBox "The value must be increased from previous row", 48, "Data check"
  grdTable.Text = ""
  Temp = ""
  GoTo dataReIn
 Else
  value1 = value2
 End If
End If

holeCounter = 0
For counter = 3 To length Step 1
 grdTable.Row = counter
 If grdTable.Text = "" Then
  MsgBox "This cell must contain data", 48, "Data Check Message"
  GoTo dataReIn
 End If
 value2 = Val(grdTable.Text)
 If value1 >= value2 Then
  MsgBox "The value must be increased from previous row", 48, "Data check"
  grdTable.Text = ""
  Temp = ""
  GoTo dataReIn
 Else
  value1 = value2
 End If
Next
 If length + 1 <= 20 Then
```

```
    grdTable.Row = length + 1
   End If
   If grdTable.Text <> "" Then
    MsgBox "The number of PS Discharge data must be less than or equal to that of Elevation
data", 48,   "Data check"
   End If
   For counter = length + 1 To 20 Step 1
    grdTable.Row = counter
    grdTable.SelStartCol = grdTable.Col
    grdTable.SelStartRow = grdTable.Row
    grdTable.SelEndCol = grdTable.Col
    grdTable.SelEndRow = grdTable.Row
    grdTable.Text = ""
   Next
  End If

  hole = False
  holeCounter = 0
  'In case B etc. there are six columns. AS Discharge field needed.
  If ascharge = True Then
   grdTable.Col = 5
   grdTable.Row = 1
   value1 = Val(grdTable.Text)
   grdTable.Row = 2
   If grdTable.Text = "" Then
    MsgBox "This cell must contain data", 48, "Data Check Message"
    GoTo dataReIn
   Else
    value2 = Val(grdTable.Text)
    If value1 >= value2 Then
```

```
    MsgBox "The value must be increased from previous row", 48, "Data check"
    grdTable.Text = ""
    Temp = ""
    GoTo dataReIn
  Else
    value1 = value2
  End If
End If

holeCounter = 0
For counter = 3 To length Step 1
  grdTable.Row = counter
  If grdTable.Text = "" Then
    MsgBox "This cell must contain data", 48, "Data Check Message"
    GoTo dataReIn
  End If
  value2 = Val(grdTable.Text)
  If value1 >= value2 Then
    MsgBox "The value must be increased from previous row", 48, "Data check"
    grdTable.Text = ""
    Temp = ""
    GoTo dataReIn
  Else
    value1 = value2
  End If

Next
If length + 1 <= 20 Then
  grdTable.Row = length + 1
End If
```

```
    If grdTable.Text <> "" Then
      MsgBox "The number of AS Discharge data must be less than or equal to that of Elevation
data", 48, "Data check"
    End If
    For counter = length + 1 To 20 Step 1
      grdTable.Row = counter
      grdTable.SelStartCol = grdTable.Col
      grdTable.SelStartRow = grdTable.Row
      grdTable.SelEndCol = grdTable.Col
      grdTable.SelEndRow = grdTable.Row
      grdTable.Text = ""
    Next
  End If

'In case the data input are acceptable, advance to
'next screen according to specific cases.
frmStructureTable.Hide
If InStr(ComString, DAMS2.Record) <> 0 Then
  frmWatershed.Show
ElseIf DAMS2.Record = "G" Or DAMS2.Record = "J" Then
  frmStrInHydro.Show
ElseIf DAMS2.Record = "H" Then
  frmWatershed.Show
End If
Exit Sub

'In case the data in the table are not correct, set
'focus to the cell and let the user input data again
dataReIn:
  grdTable.SelStartCol = grdTable.Col
```

```
    grdTable.SelStartRow = grdTable.Row
    grdTable.SelEndCol = grdTable.Col
    grdTable.SelEndRow = grdTable.Row
    grdTable.SetFocus
End Sub
```

Figure 20. Screen Level Data Check Example

All the screens except the final one in this project contain a "Next Screen" command button. The "Click" message handling procedure accomplishes the screen level data validation check task.

4.2.3 Application level data validation

The field level and screen level data checks handle most part of the data validation. In some cases the data in a specific screen not only need obey the rules of the data themselves, such as data type, maximum and minimum value, etc., but also are dependent on other data or some specifications in other screens. In this case, the field and screen level data checks are not sufficient. We need application level data validation to accomplish this check.

I did not implement the application level data validation as a separate part. Instead, I built it into the field and screen level data check modules. This means when a field or screen level data check is in progress, any data that require cross screen data check will be done in the same procedure. I defined several global variables in this project. Some of these global variables were used to help the application level data check.

For example, the application has two global variables, "maxElev" and "minElev", which were used to hold the last and first input values for the "STRUCTURE" table "Elevation" column. The future "Auxiliary Spillway (AS) Crest" screen has a field for an "ESCREST" record that requires the input value to be less than the "maxElev" and greater than or equal to the "minElev" values. This part of the source code is listed below:

```
Private Sub txtAux1_LostFocus()
 Dim value
 Dim OutRange As Boolean

 OutRange = False
 If txtaux1.Text <> "" Then
  value = Val(txtaux1.Text)
  If option2 = True Then
   If value >= maxElev Or value < minElev Then
    MsgBox "The Crest must be less than " + Str(maxElev) + " and greater than or equal to "
+          Str(minElev), 48, "Data Check"
    OutRange = True
    txtaux1.Text = ""
   End If
  Else
   If value < 0 Then
    MsgBox "The Crest must be greater than or equal to 0, try again.", 48, "Data Check"
    OutRange = True
    txtaux1.Text = ""
   End If
  End If
 End If
```

```
    If Enter = True And OutRange = False Then
      Enter = False
      If txtaux2.Visible = True Then
        txtaux2.SetFocus
      Else
        cmdNext.SetFocus
      End If
    End If
End Sub
```

Figure 21. Screen Level Data Check Example

This example shows how I used global variables to help implement the application level data check. Actually, this is not the only way to do across screen data check. In many situations, I used screen control contained data values directly to do this kind of check. The global variables were used only in situations where I could not directly use other screen data to do a data check. Global variables are not good for system modulation. It is better to avoid using many global variables.

4.3 Control File Management

The goal of the project is to generate and manage the control file. The current version of the project contains a single menu item "File". The "File" menu has six entries: "new", "Open", "Continue", "Save", "Save As", and "Exit". These menu entries were used to manage the control file system.

4.3.1 New Control File

If a user selects the "New" entry from "File" menu, a new control file generating process is started. Almost every screen in the project has a "Home Screen" button that could lead the user back to the program starting screen that contains the "File" menu. We need different solutions for different situations.

In the case of a user already having a file opened before he or she chooses the "New" in "File" menu again, the solution is this: First the program will check if the open file has anything changed since the file was opened. If nothing is changed, the new file editing process will continue normally. Otherwise, if the open file is changed, a dialog box will pop up to prompt the user to save the file before creating a new file. If the user answers "yes", the program will save the changed file under its original file name, then continue with the new file editing process. If the user answers "No", the program will do nothing to the current open file and will let the user work on the new file.

The program does not support multiple file creation. If the user already has a new file in progress when he tries to create another new control file, I provided another solution. First a dialog box pops up to ask the user if he wants to save the file before creating another new file. If the user answers "Yes", the "Save As" routine will be called. After the file is saved, the new file creating process continues. If the user answers "No," the information which the user inputted for the previous "New" file will be lost, and another "New" file creating process begins.

The new file creating process will lead the user through a set of screens to input information. When the user reaches the last screen, he can directly click the "Save" button to save the control file to disk.

4.3.2 Open Existing File

A user can choose "Open" in the "File" menu. The "Open" entry has a handling procedure that helps the user to open an existing SITES control file. Like the "New" entry selection, the "Open" entry also need different solutions for different situations.

If the user already has a file open before he selects the "Open" entry, the solution is: If the previously opened file has been modified, first prompt the user to save the previously opened file, then either save the file or do nothing according to the user's response. After the housekeeping finishes, the Open file process begins.

If the user has a new file creating process started before he clicks the "Open" entry in the "File" menu, we need do the same thing as in "New" entry selection. After that, the user will enter the open file session.

The open file process will load the control file that the user selects into a set of screens for a specific path. In this process, the user can go back and forth in the path, and change any data in the file. The user also can change path from the current loaded file. Before the user selects "Save", or "Save As" entry in "File" menu, or "Save" button in the final screen, nothing is really changed from the original control file.

4.3.3 Continue Working on the File

The "File" menu entry, "Continue," is designed for a special situation. When a user already has opened a file or is in the middle of a new file creating process, a push of the "Home Screen" in any one of the screens will take the user back to the starting screen. In this screen, the user has several possible selections from the "File" menu: creating another new file, opening another file, saving the file, and exiting the program. But there is a case when the user just wants to continue working on the file he or she has already opened or started creating. The "Continue" entry is provided for this purpose.

When a user initially starts the program, the "Continue" entry in the "File" menu is not enabled. It is enabled only after the user opens or starts a new file and jumps back to the start screen. After the user saves the control file, the "Continue" entry will go back to the disabled state.

4.3.4 Save Control File

A user can save his or her control file to disk from three places. The "Save" and "Save As" entries from "File" menu in the program starting screen let the user save the control file from in any point of the program. The user can push the "Home Screen" button from most of the screens to go to the program start screen and select "Save" or "Save As" menu entry from that screen. The file generated from these two menu selections may be an incomplete control file which cannot be used for actual design and analysis.

Another place where the user can save the control file is in the last screen for a file generating path. That screen contains a "Save" button to allow the user to save the file with a selected file name. This screen only appears when the user has finished all the required data input. The control file generated from the last screen "Save" button is a complete control file which can be used to run a design and analysis in SITES.

The Windows standards for "Save" and "Save As" menu selection were applied in this application. When a new file generating process starts, there is no difference for the first time user to choose "Save" or "Save As". After first-time saving when the file already has a file name, when the user chooses "Save" again, the file will directly be saved to the existing file name. If the user chooses "Save As", a standard Windows file save dialog box will pop up to let the user type in a file name under which to save the control file.

4.3.5 Exit from the program

Users can choose the "Exit" entry from the "File" menu to exit the program. There are housekeeping jobs for the program to do before shutting down the program.

If there are any files opened and modified but not saved yet, or a file creating session has started but the file has not been saved yet, the "Exit" handling procedure will prompt the user to save the file before closing the program.

The "Exit" handling procedure is also responsible for unloading all the forms and freeing all the resources the current process has been using. Using "Exit" to exit from the program ensures the system is in a safe state after the program has finished.

4.4 Windows Help System

The project contains a Windows standard help system. The client creates the help file. I used the Windows help authoring tool "Help Magician Pro" to build the standard Windows help system. The help system is on the screen level. The user invokes help topics by clicking the "Help" button on each screen. All the data fields are included in the help topic. The user also can search for help information by typing keywords on the main help screen.

CHAPTER 5

CONCLUSION

The SITES control file generation interface program is a part of the series of SITES products. A successful control file interface may win more users for the products and it may also help increase SITES engineers' productivity.

The goal of this part of the project is make data input operations "User friendly" and "Correct".

To achieve these goals, I always kept users's needs in mind in the interface design and implementation. The resulted interface meets the needs of users of any level.

This version of the control file interface also can run on several Windows system. The single document interface (SDI) approach and the multiple path screens combination design enable the program to require less system resources when it is running. Users who still use old machines such as an Intel 80386 PC can also run this version of the program.

The three levels (data field, screen, application) of integrated data checks in the program help users generate a correct control file used for real world design and analysis tasks. The interface menu system helps users safely manage the control file

generation process. The standard window on-line help system provides enough information to assist users when they are generating the control file.

This is the first version of a SITES control file generating program for Windows. This version only supports nine "Cases" of designing control files. It is possible that more "Cases" will be identified when the new designing or analysis requirements comes out. In recognition of this possibility, I built some space in the interface design for future development. If one adopts the same approach used in developing this version, it will not be very difficult to plug in some more "Cases" into the current program module.

This version of the program can only be used to generate a control file which contains single-run information. The next step is to develop the multi-run control file generating interface. Based on this version, the "control file append" function can be added to generate a multi-run control file.

When designing and implementing this program, I attempted to make the program upgradable. Dam design and analysis, the topics the application deals with, are under constant research. Along with increasingly advanced programming techniques, I expect to see more sophisticated versions in the future.

BIBLIOGRAPHY

D.M. Temple, H.H. Richardson, J.A. Brevard, G.J. Hanson (1995). *SITES: The New Dams2.* Applied Engineering in Agriculture. 1995 American Society of Agriculture Engineers. Vol. 11 (6): 831-834

D.M. Temple, G.J.Hanson (1994). *Headcut Development in Vegetated Earth Spillways.*

Applied Engineering in Agriculture. 1994 American Society of Agriculture Engineers. Vol. 10 (5): 677-682

D.M. Temple, J.S. Moore, (1994) *Headcut Advance Prediction for Earth Spillway.* Presentation at the 1994 ASAE International Winter meeting

D.M. Temple, J.A. Brevard, J.S. Moore, G.J. Hanson, E.H. Grissinger, and J.M. Bradford. (1993) *Analysis of Vegetated Earth Spillways.* Proceedings of Transctions of 10th Annual Conference, The Association of State Dam Safety Office.

Kansas State Board of Agriculture Division of Water Resource (1986) *Engineering Guide -1 EG -1.*

Kansas State Board of Agriculture Division of Water Resource (1986) *Engineering Guide -2 EG -2.*

Elisabeth Boonin, (1996) *User Interface Design.* Visual Basic 4 Expert Solutions. Que Corporation, Indianapolis, IN.

Steve Potts, (1996) *Multiple Document Interface*. Visual Basic 4 Expert Solutions. Que Corporation, Indianapolis, IN.

Steve Potts, (1996) *Optimizing VB Code*. Visual Basic 4 Expert Solutions. Que Corporation, Indianapolis, IN.

Jon Oelschlaeger, (1996) *Developing Online Help.* Visual Basic 4 Expert Solutions. Que Corporation, Indianapolis, IN.

S. Rama Ramachandran, (1996) *Using the Windows API.* Visual Basic 4 Expert Solutions. Que Corporation, Indianapolis, IN

Microsoft Corporation, *Macrosoft Visual Basic Program's Guide*. (1996) Microsoft Corporation

Microsoft Corporation, *Macrosoft Visual Basic Language Reference*. (1996) Microsoft Corporation

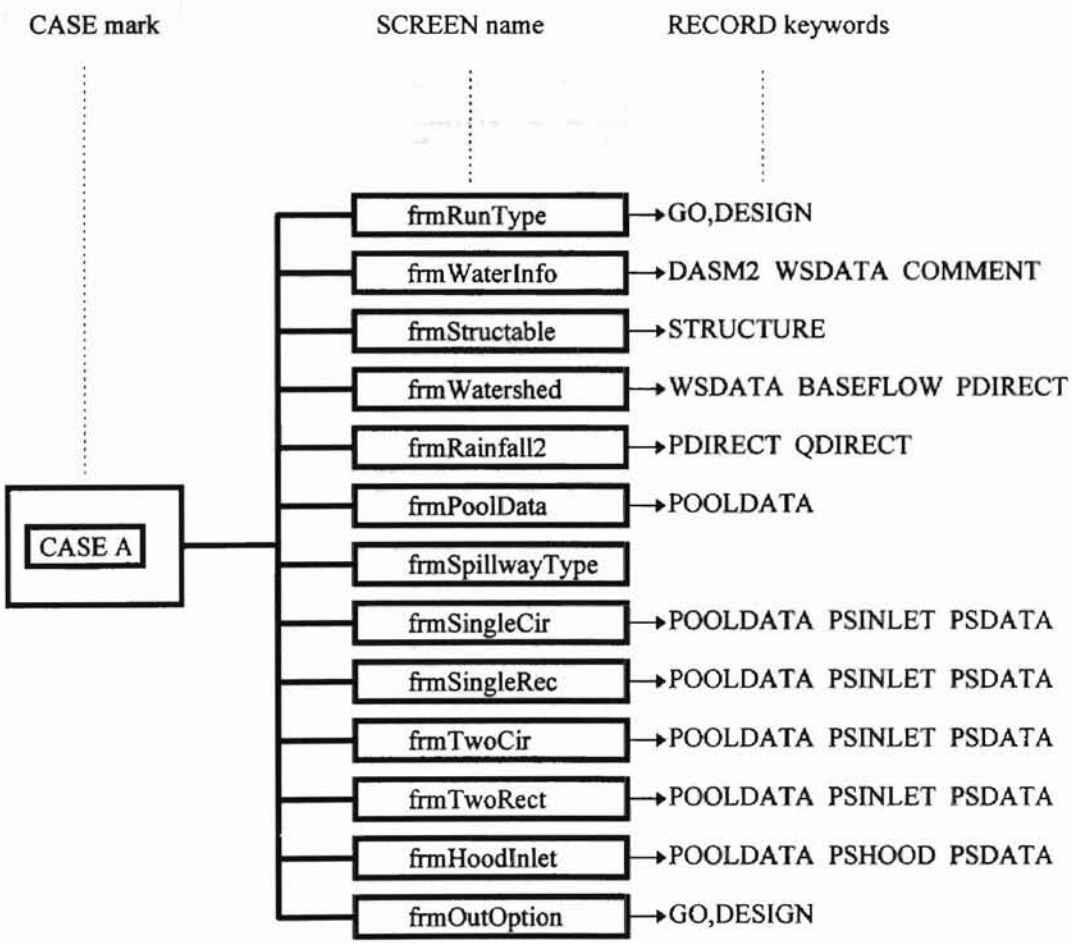
Robert B. Heberger, (1995) *The Windows Help Magician Revision 3.1.* Software Interphase Incorporated

APPENDIX A

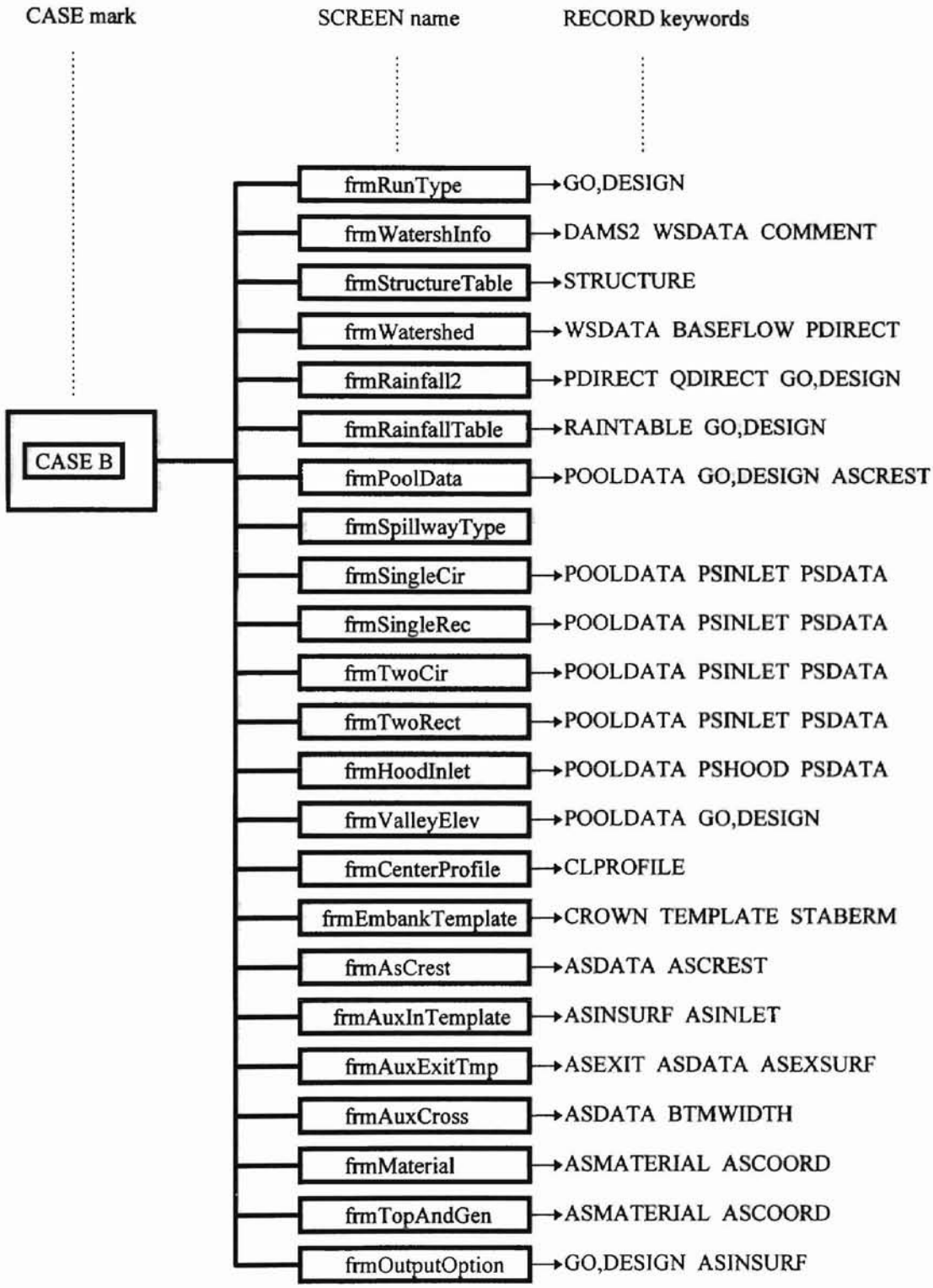
CASES, SCREENS, AND RECORDS GRAPH

This appendix includes the nine Cases' screens and records relationship graphs. Every graph has three parts from left to right. The three parts represents "Case", visual basic screen (form) name, and records name in the screen. These graphs are useful for understanding and maintaining the structure of the project.

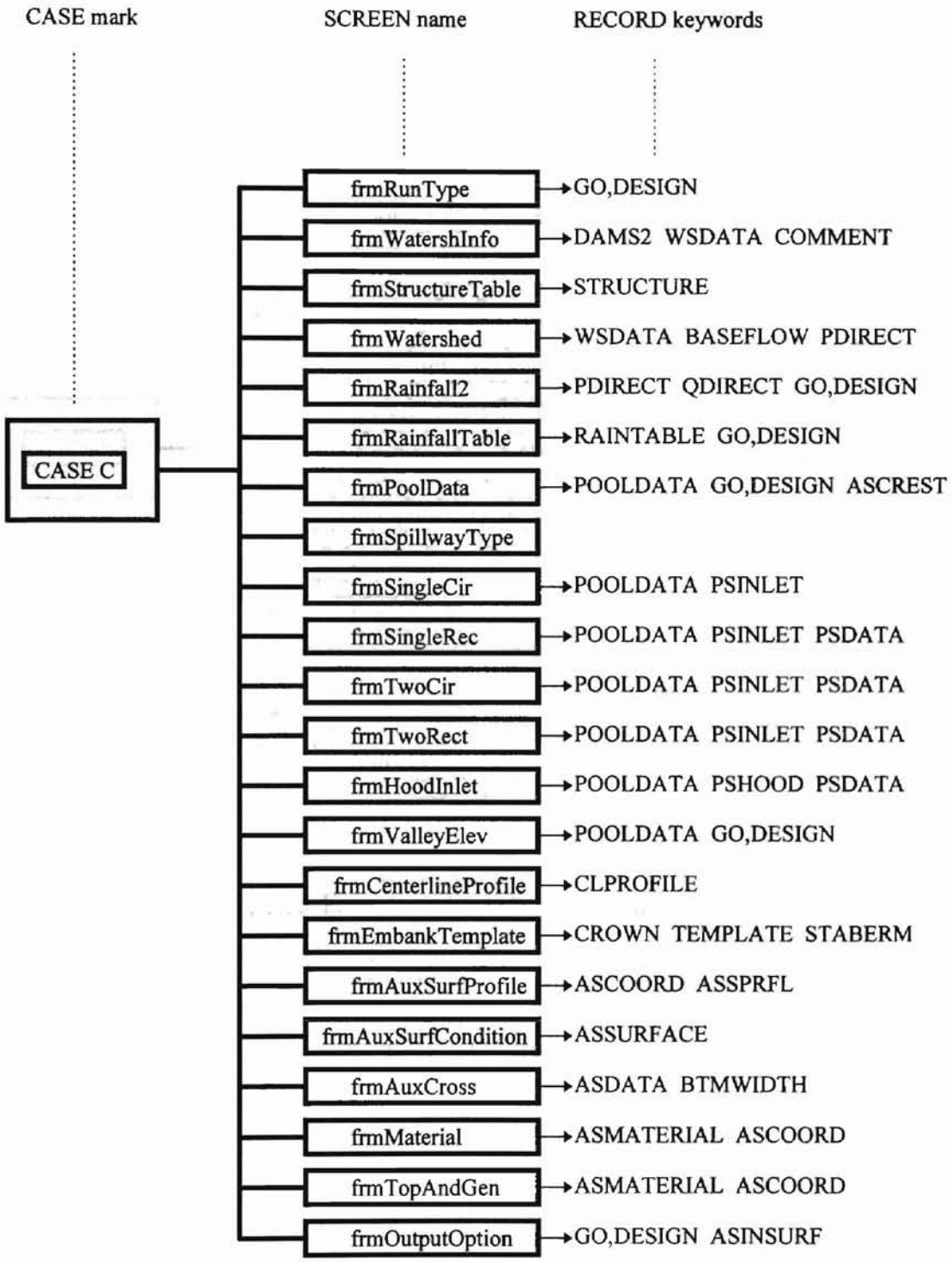
1) Case A Graph



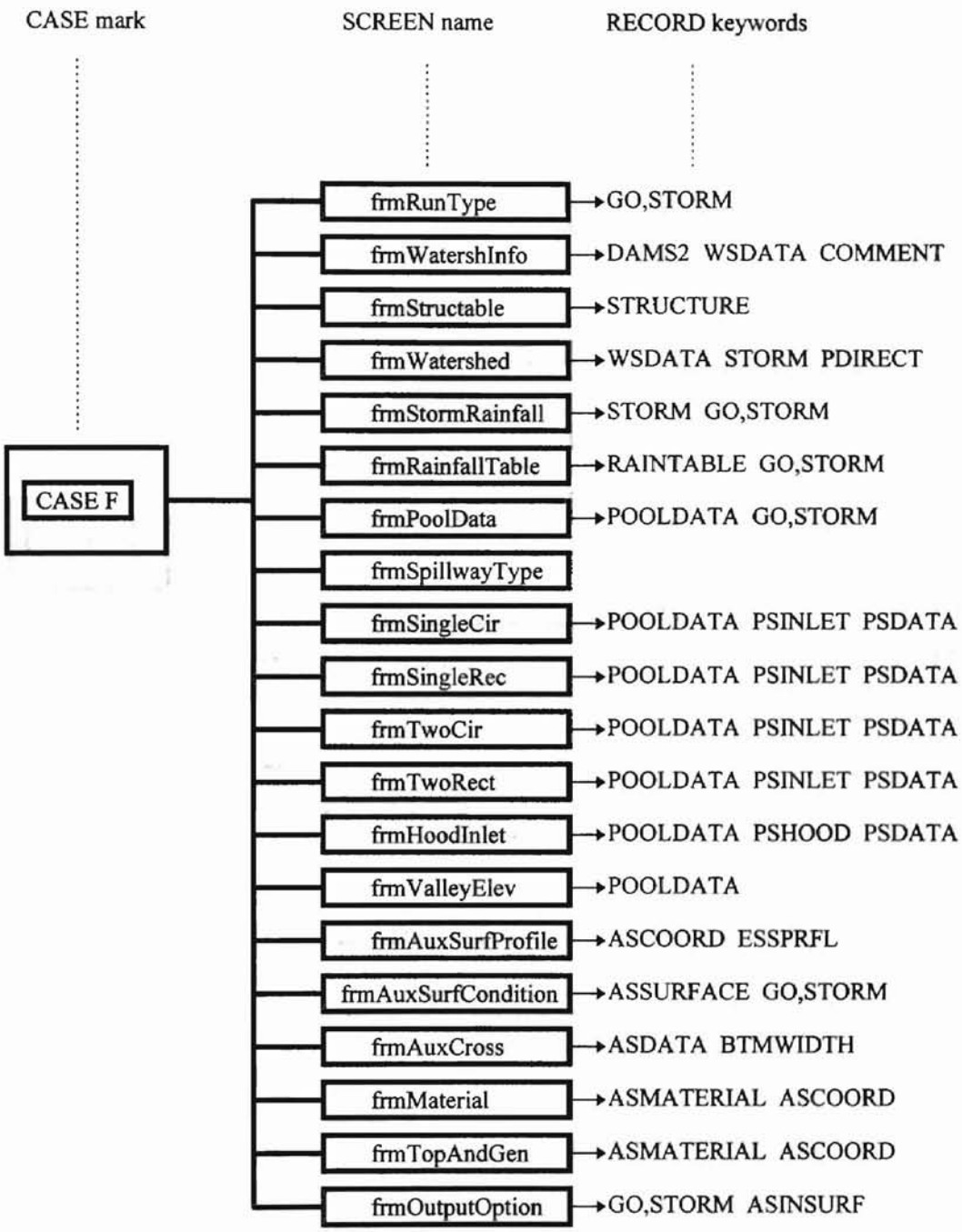
2) Case B Graph



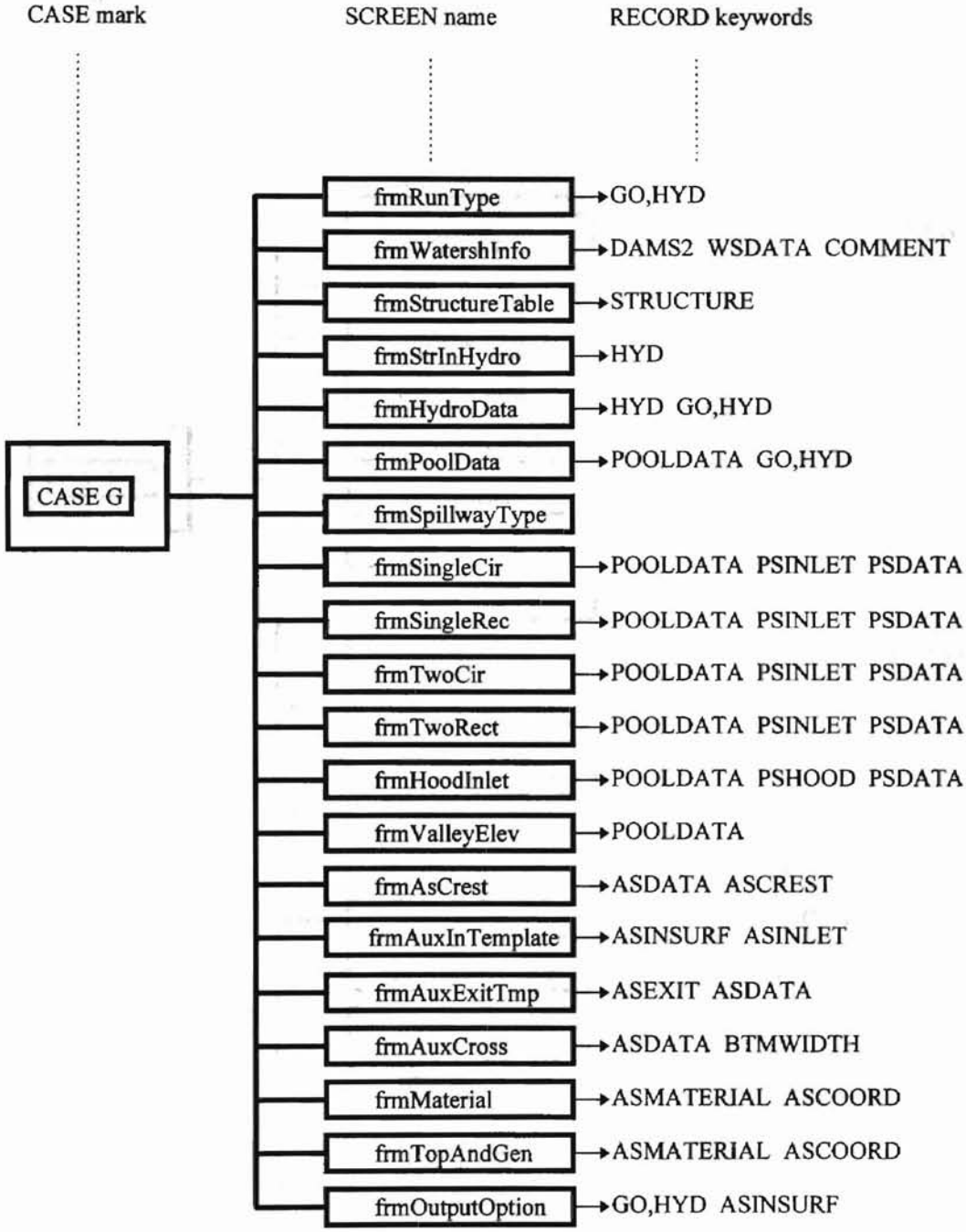
3) Case C Graph



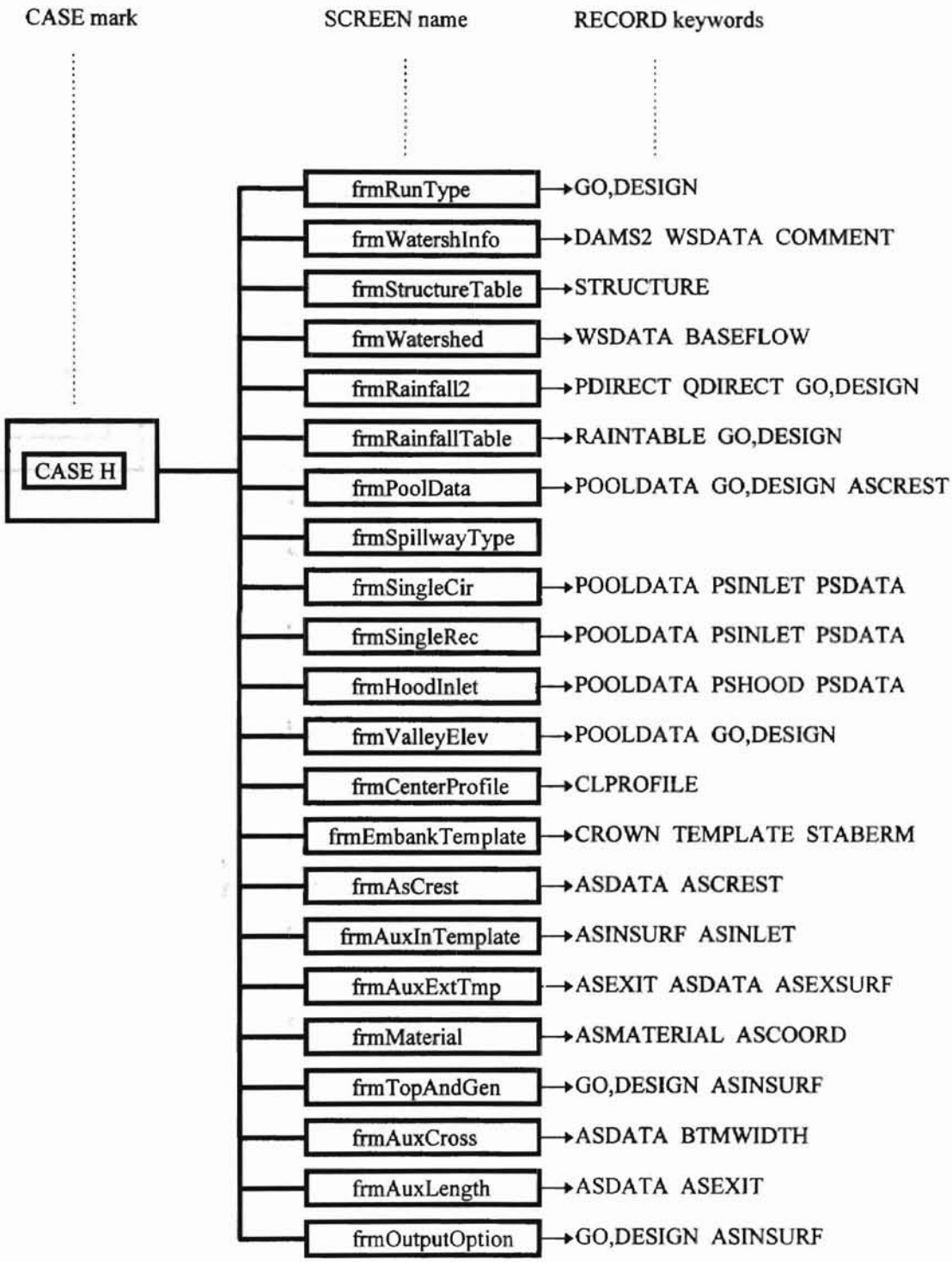
4) Case F Graph



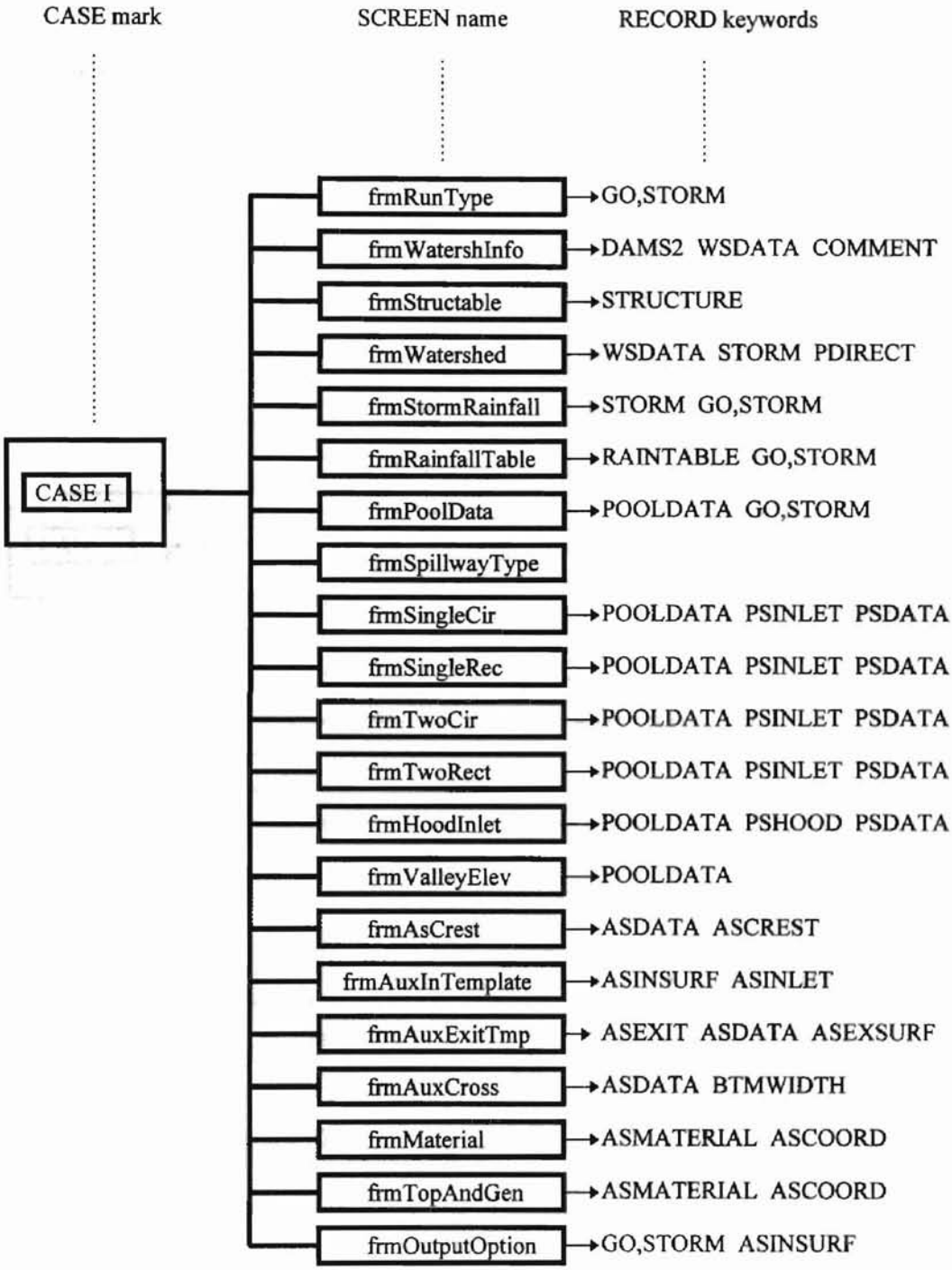
5) Case G Graph



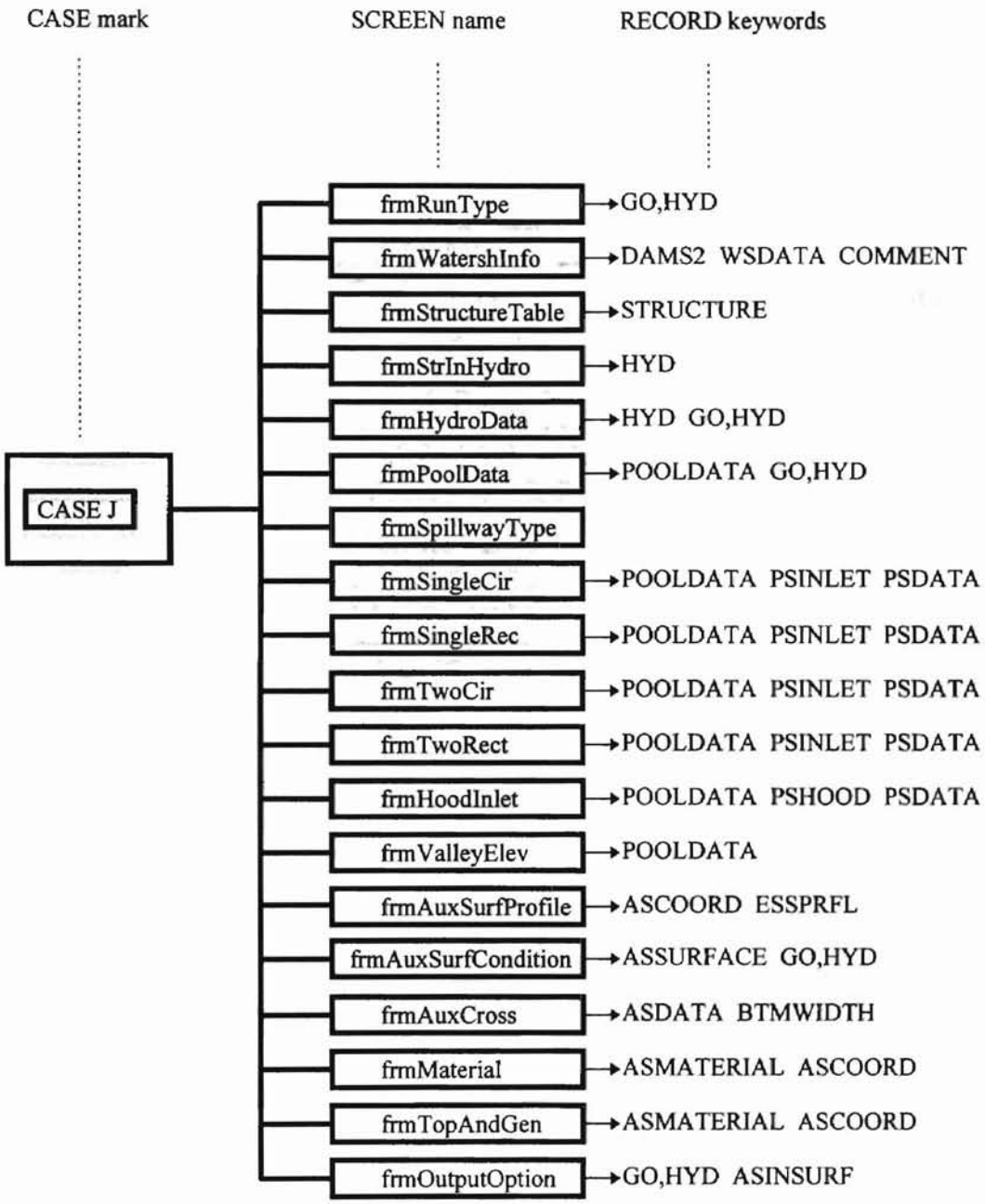
6) Case H Graph



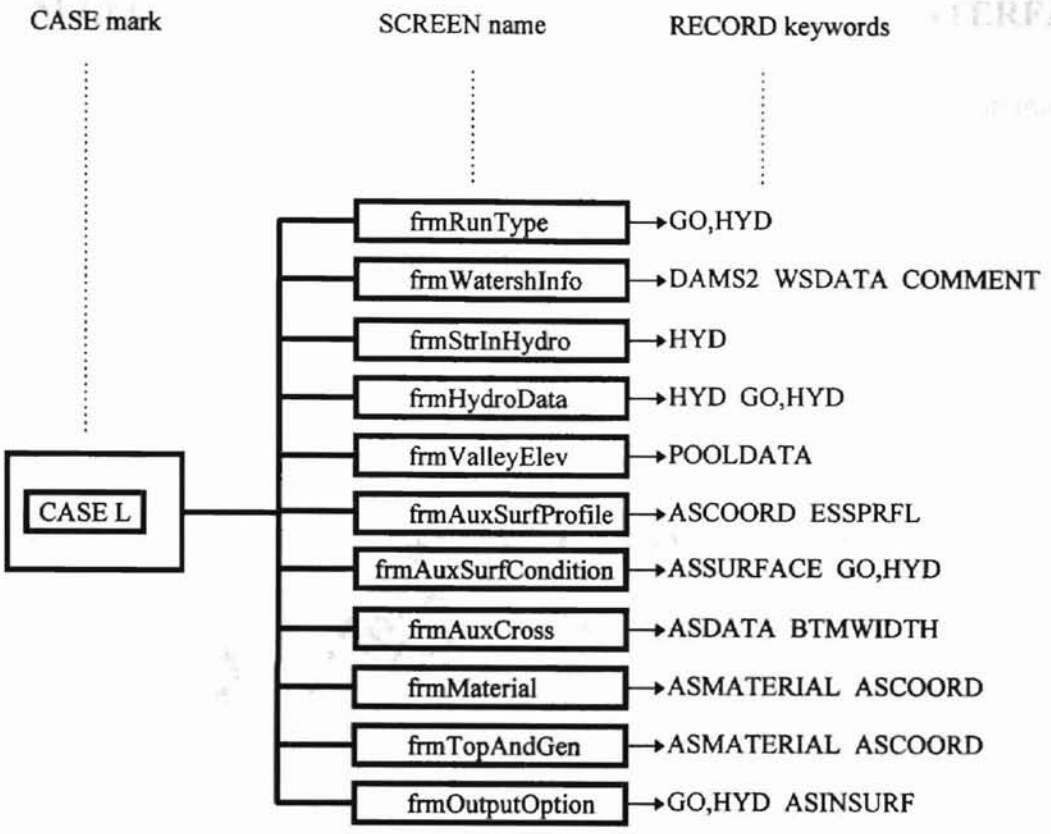
7) Case I Graph



8) Case J Graph



9) Case L Graph



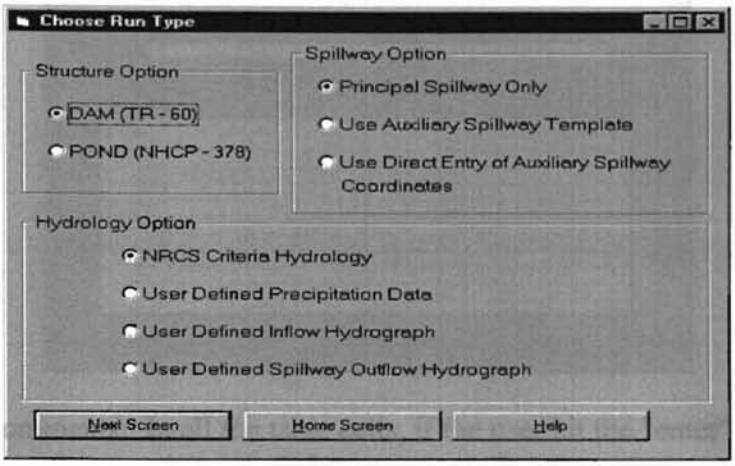
APPENDIX B

SELECTED SCREENS OF SITES CONTROL FILE INTERFACE

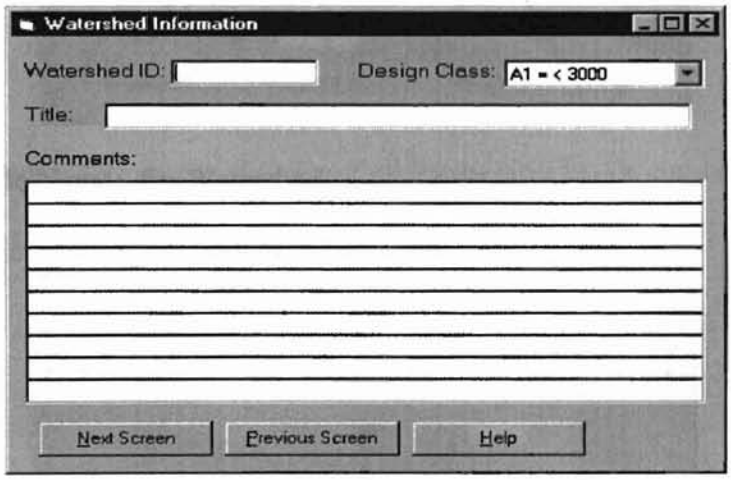
This Appendix includes "Case A" and "Case F" control file generation interface screens. The interface actually has nine different "Case". The two "Case" displayed here is for demonstrating the screen designing.

1.) Case A

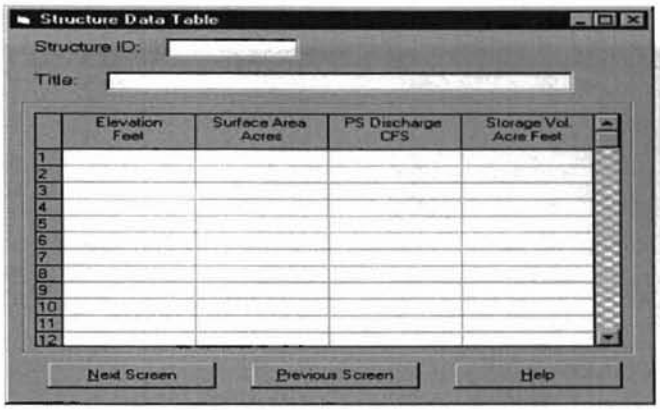
Comments: This is the first screen when the program starts. The user can choose "new" or "open" from the "file" menu. "new" means start to build a new control file. "open" means open an existing control file.
File name: frmSite



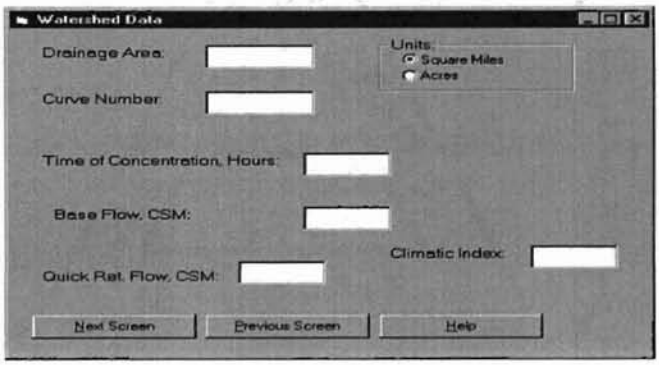
Comments: This is the first screen for control file editing. Any cases of control file should start from here. Radio buttons provide the selections for different cases.
File Name: frmRunType



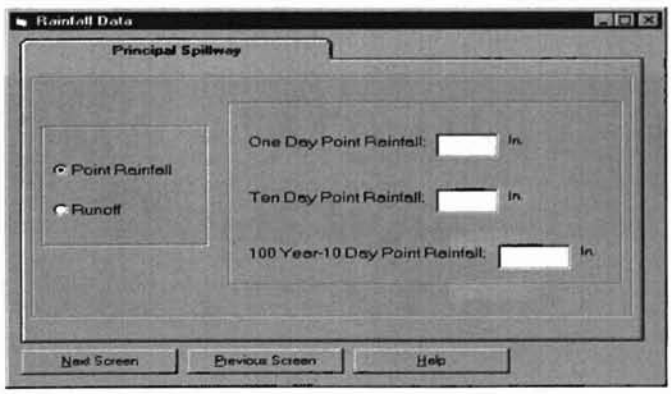
Comments: Design class is not a user input field. The user should choose from the push-down combo box.
File Name: frmWatershInfo



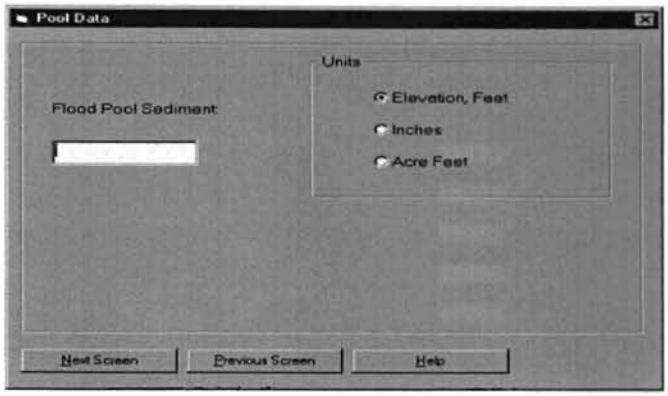
Comments: In all the table cells, if the user hit the "enter" key after entering value, the program will automatically check data validation. Otherwise the integrated data check will be done when the user push "Next Screen".
File Name: frmStructureTable



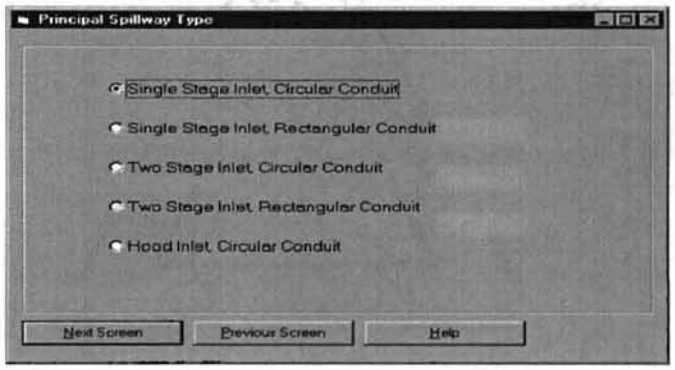
File Name: frmWatershed



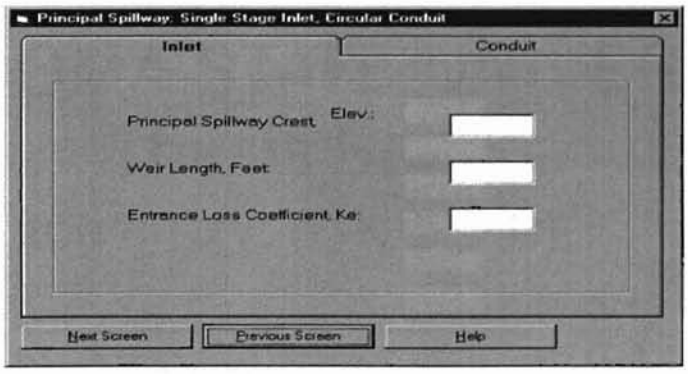
Comments: This a run time dynamic change screen for different cases. When a user change path (case), the new fold will be added.
File Name: frmRainfall2



Comments: This is also a run time change screen for different cases.
File Name: frmPoolData

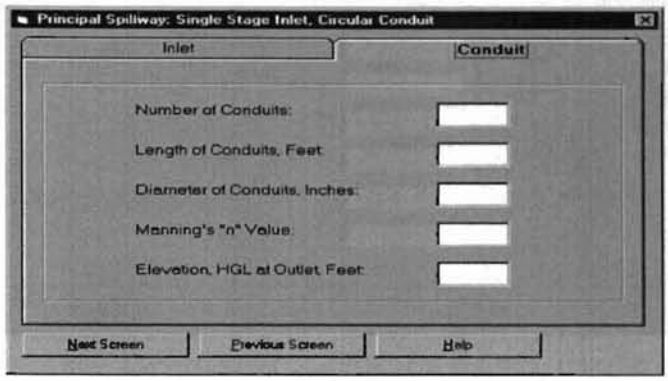


Comments: From this screen, different path inside one case are chosen.
File Name: frmSpillwayType

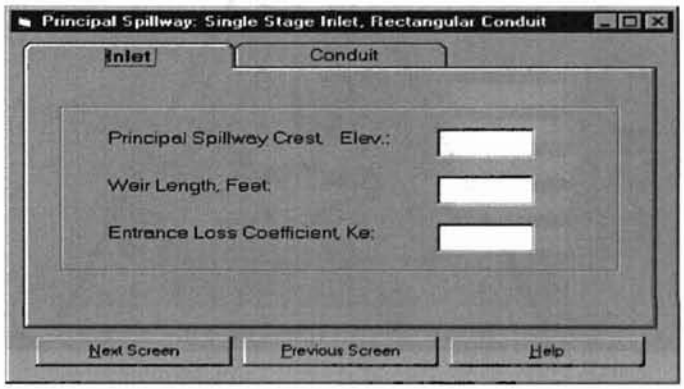


Comments: This is the first selection path from the previous screen. It has two folders. This is one of the folders.

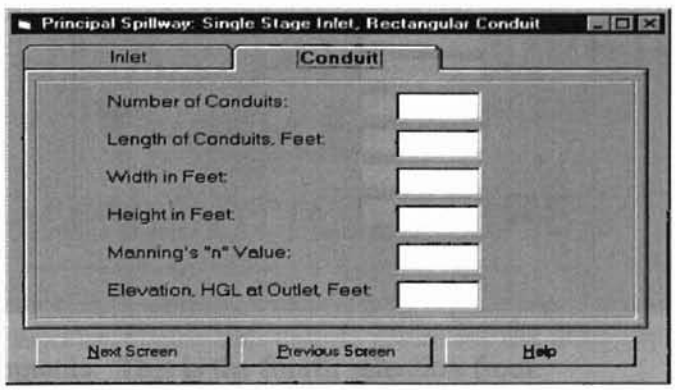
File Name: frmSingleCir



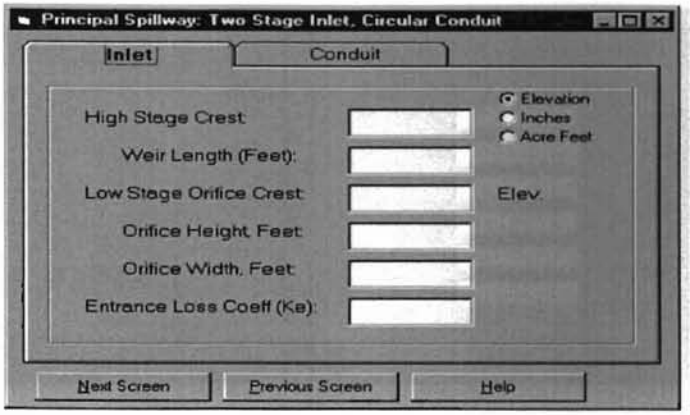
Comments: This is another folder in the same screen with previous screen.
File Name: frmSingleCir



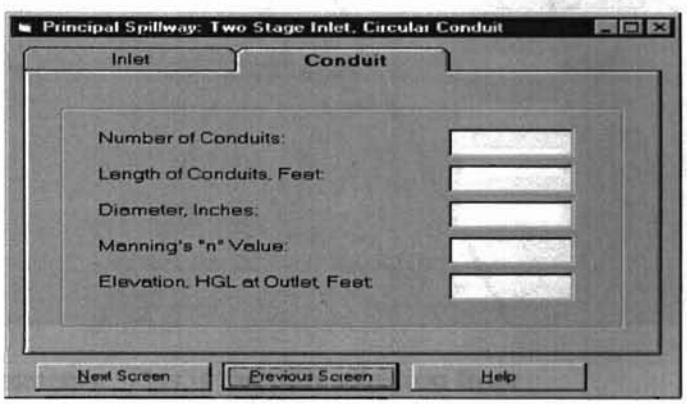
Comments: This is the second selection from "frmSpillwayType".
File Name: frmSingleRec



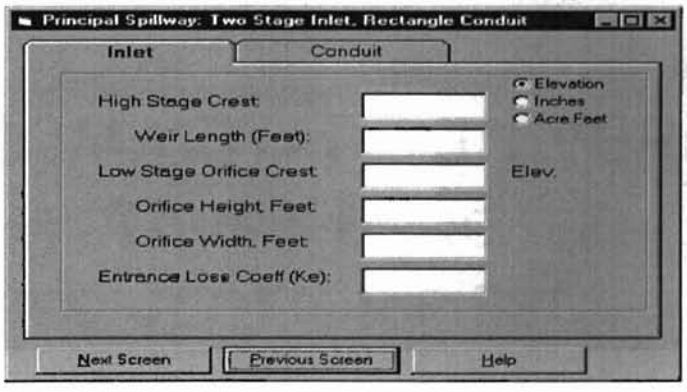
Comments: This is another folder for the previous screen.
File Name: frmSingleRec



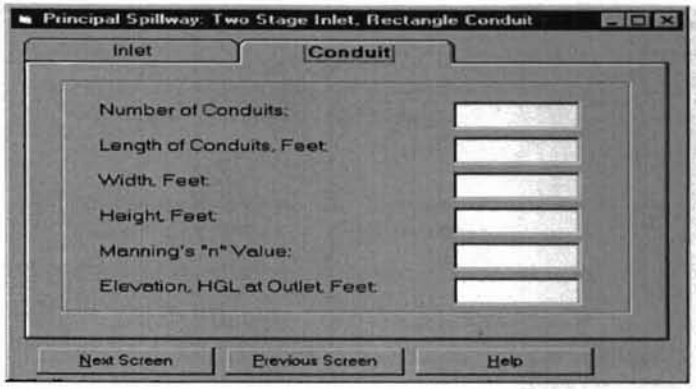
Comments: This is the third selection path from "frmSpillwayType".
File Name: frmTwoCir



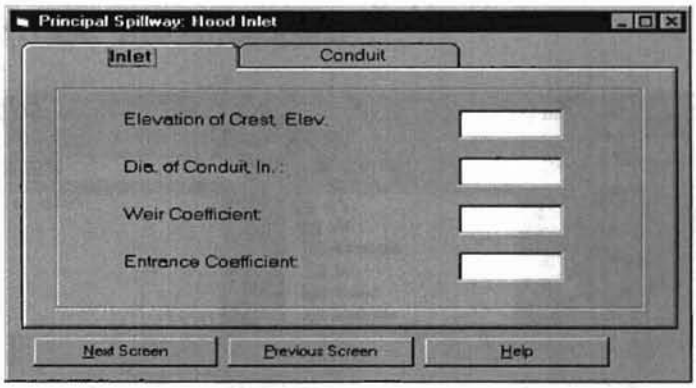
Comments: This is another folder of the previous screen.
File Name: frmTwoCir



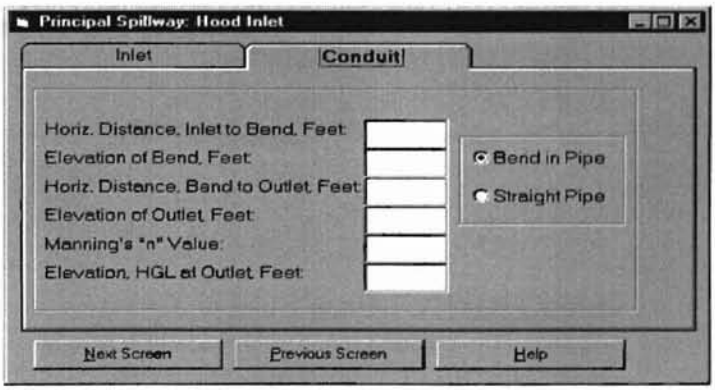
Comments: This is the fourth selection from "frmSpillwayType".
File Name: frmTwoRect



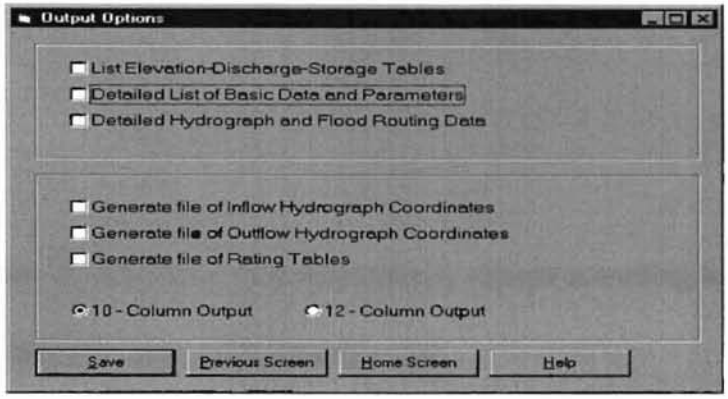
Comments: This is another folder for the previous screen.
File Name: frmTwoRect



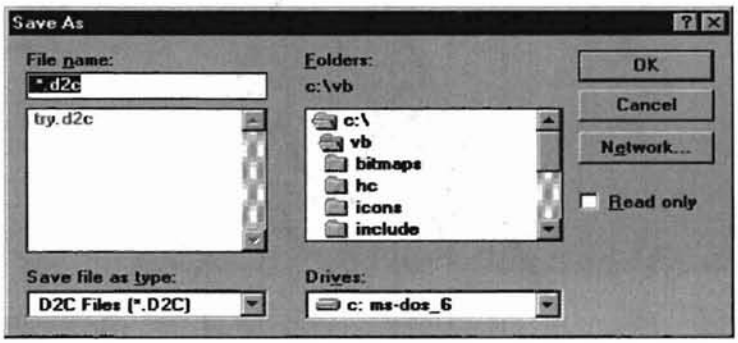
Comments: This is the fifth selection from "frmSpillwayType".
File Name: frmHood



Comments: This is the another folder for the previous screen.
File Name: frmHood



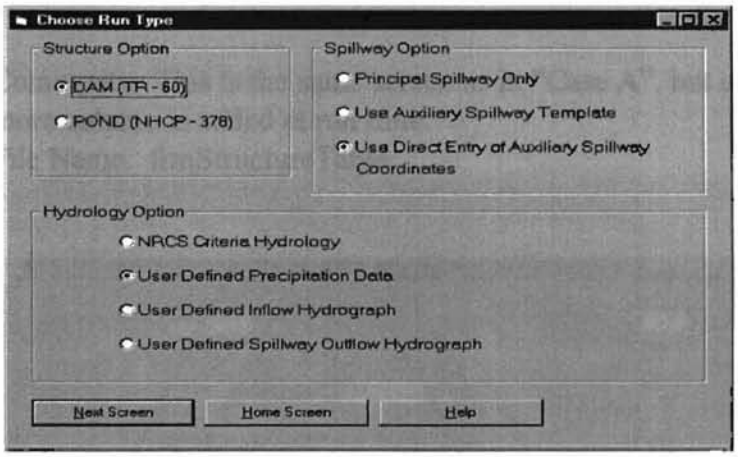
Comments: This is the final screen for a Case A run. A user can click the "Save" button to the control file. When Save is clicked, a file save dialog appears.
File Name: frmOutputOption



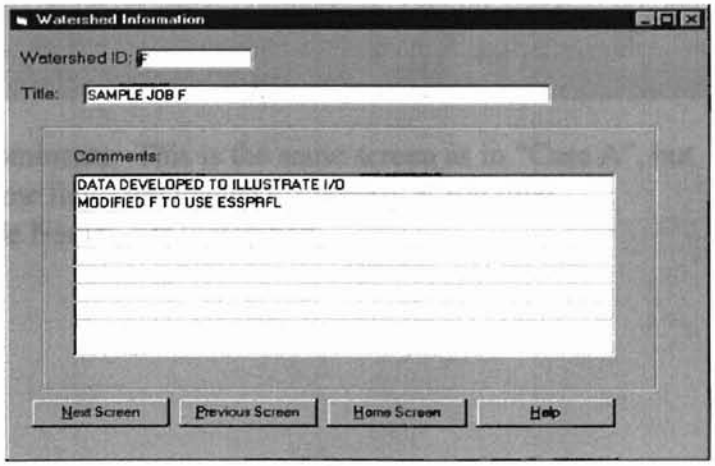
Comments: This is file selection dialog box. When a previous "Save" button has been pushed, this will pop up. It is not a single file. It is a common dialog box built into Visual Basic program.

2.) Case F

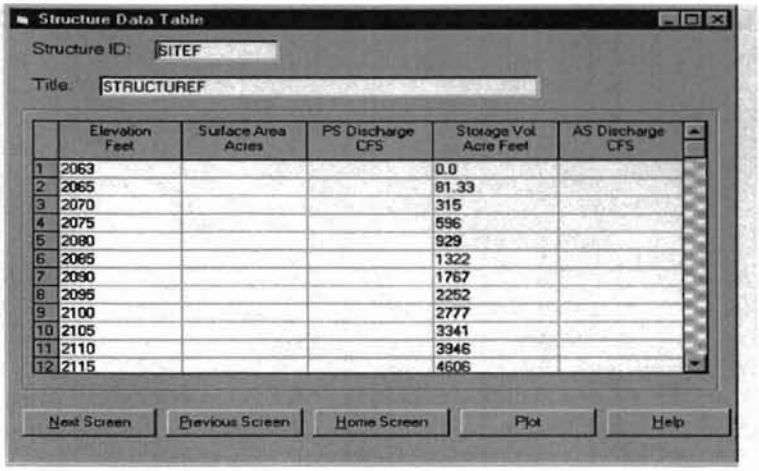
Case F has several screen may dynamically change according to a user's selection. Here I pick one possible path screens for this case.



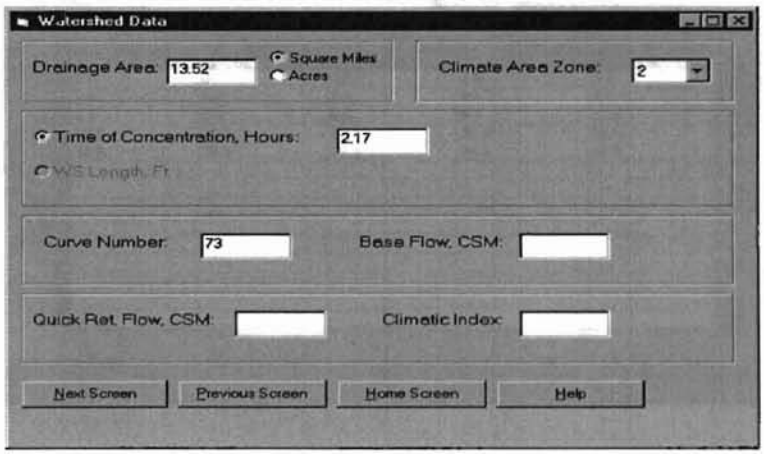
File Name: frmRunType



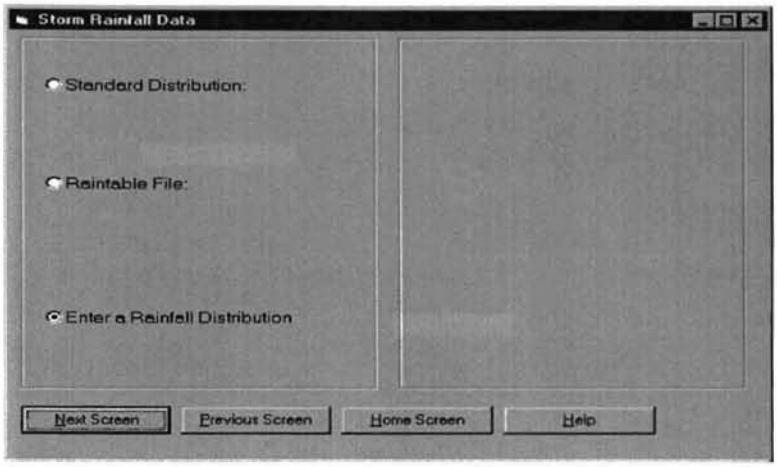
File Name: frmWatershInfo



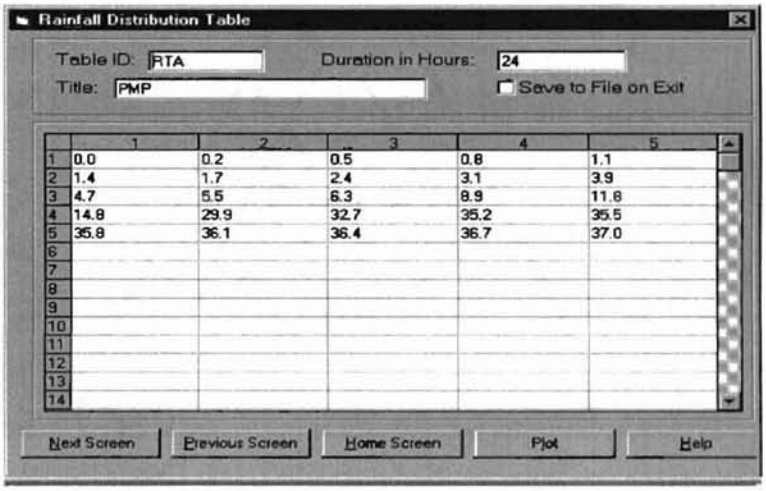
Comments: This is the same screen as in "Case A", but one more column is added at run time.
File Name: frmStructureTable



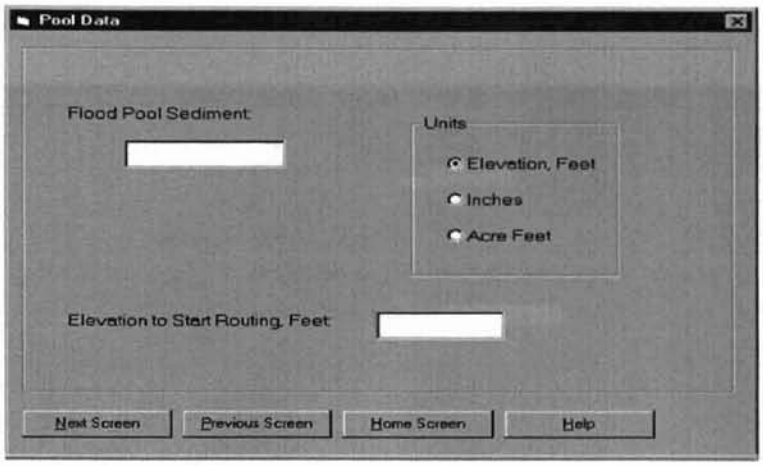
Comments: This is the same screen as in "Case A", but some fields are loaded differently at run time.
File Name: frmWatershed



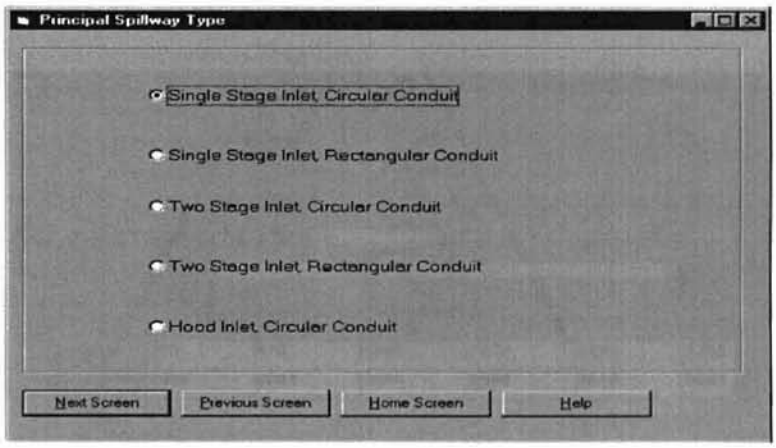
File Name: frmStormRainfall.



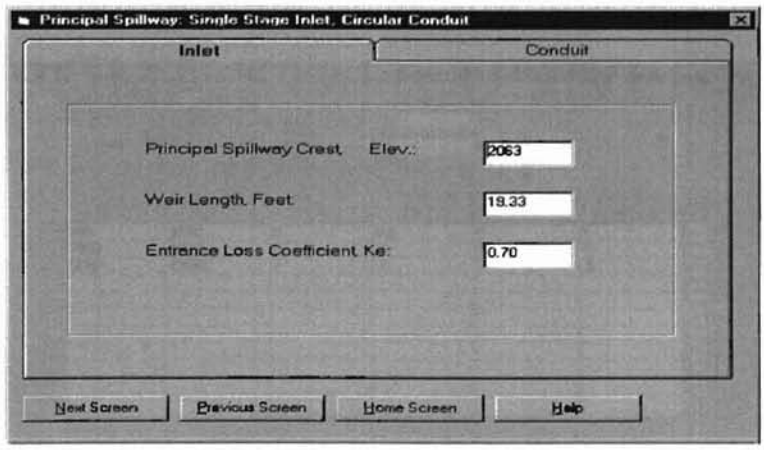
File Name: frmRainfallTable.



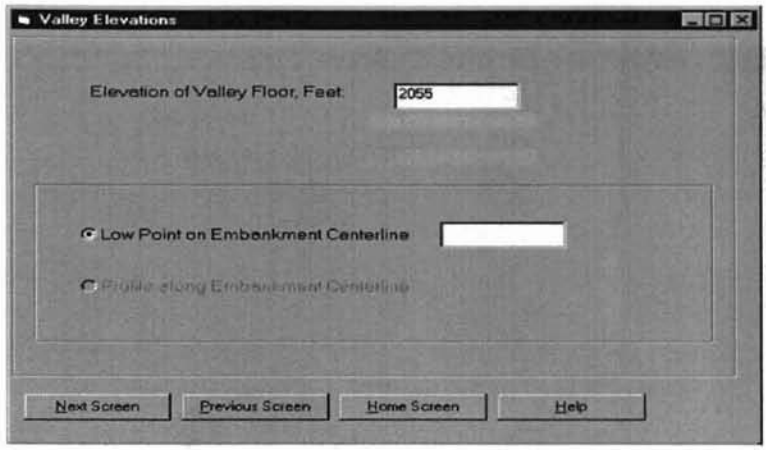
File Name: frmPoolData.



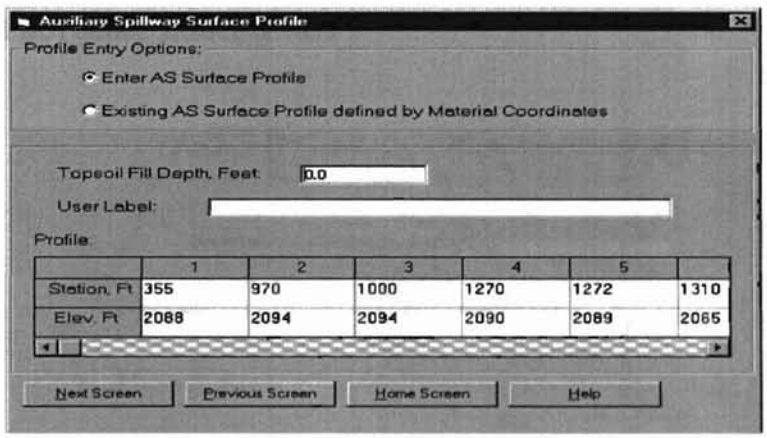
File Name: frmSpillwayType



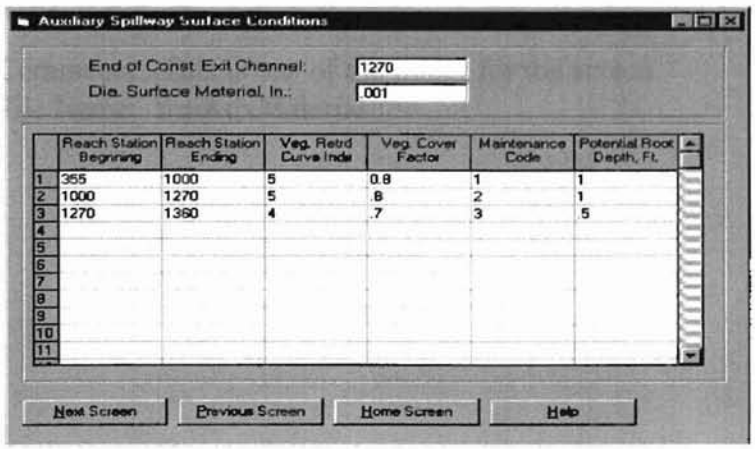
File Name: frmSingleCir



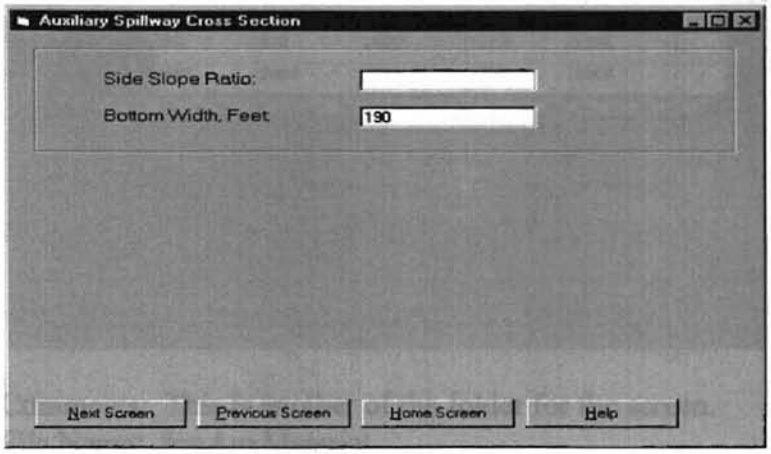
File Name: frmValleyElev



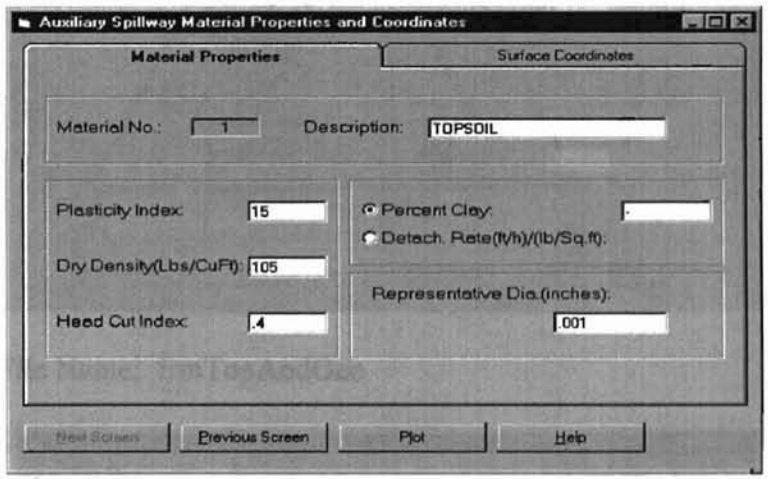
File Name: frmAuxSurfProfile



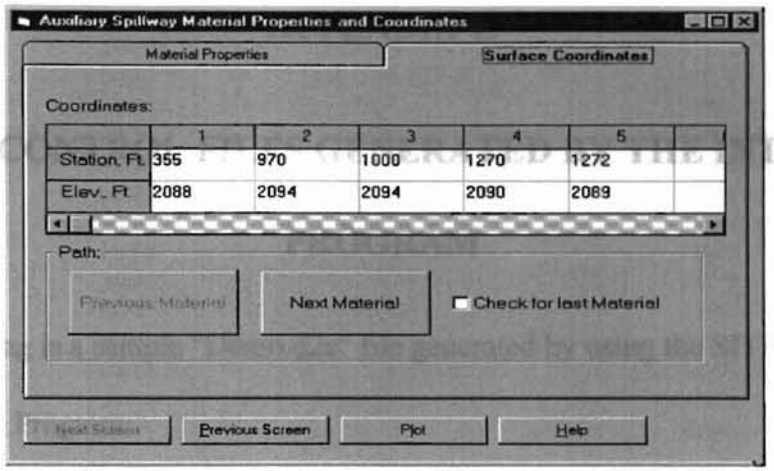
File Name: frmAuxSurfCondition



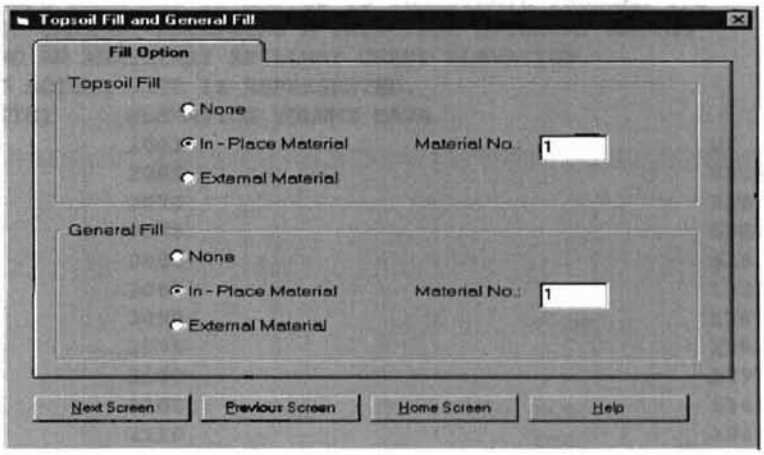
Comments: This screen will dynamically change according to different cases at run time.
File Name: frmAuxCross



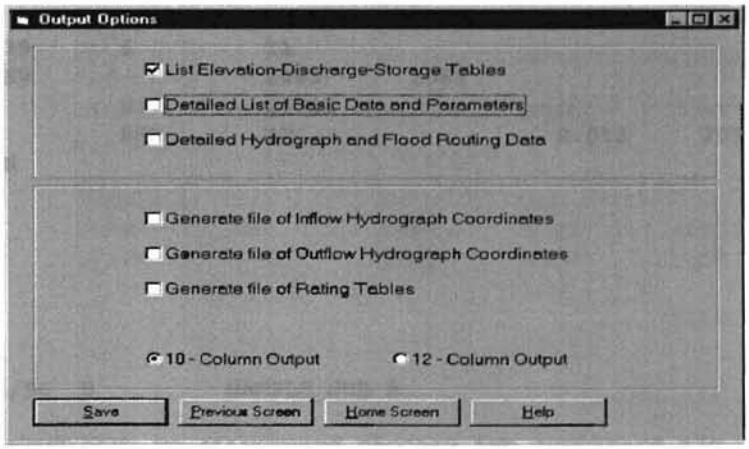
Comments: This is one of the folder for the screen.
File Name: frmAuxMaterial



Comments: This is another of the folder for the screen.
File Name: frmAuxMaterial



File Name: frmTopAndGen



File Name: frmOutputOption

APPENDIX C

SAMPLE CONTROL FILES GENERATED BY THE INTERFACE PROGRAM

Following is a sample "Demo.d2c" file generated by using the SITES control file Interface Program:

1) Case A

```
DAMS2      03/01/95  A          CASE A SAMPLE JOB                          A
*
*          THESE DATA ARE DEVELOPED TO ILLUSTRATE PROGRAM I/O
*          FOR CASE A: DETERMINE A PRINCIPLE SPILLWAY RATING
*          AND AN AUXILIARY SPILLWAY CREST ELEVATION.
*          NO ACTUAL SITE IS REPRESENTED.
STRUCTURE  SITE1      ELEVATION VOLUME DATA
                      2063                                    0
                      2065                                    81.33
                      2070                                    315.62
                      2075                                    596
                      2080                                    929.18
                      2085                                    1322.14
                      2090                                    1767.54
                      2095                                    2252.66
                      2100                                    2777.35
                      2105                                    3341.82
                      2110                                    3946.92
                      2115                                    4606.82
                      2116                                    4751.02
                      2117                                    4900
ENDTABLE
WSDATA     0C  A      73         13.52      2.17
PDIRECT    1.39       6          11
POOLDATA   ELEV                  2063       2063                          SC
PSINLET                0.7        19.33
PSDATA     1           560        42                    0.012      2033
GO,DESIGN  LPN
ENDJOB
ENDRUN
```

2) Case B

```
SITES      03/01/96  B          SAMPLE JOB B                               B
*
*          DATA DEVELOPED TO ILLUSTRATE PROGRAM AND INTERFACE I/O
*          AUXILIARY SPILLWAY TEMPLATE
*          NO ACTUAL SITE OR DESIGN REPRESENTED
```

```
STRUCTURE SITE1     STRUCTUREB
                    2063                                        0
                    2065                                        81.33
                    2070                                        315.62
                    2075                                        596.00
                    2080                                        929.18
                    2085                                        1322.14
                    2090                                        1767.54
                    2095                                        2252.66
                    2100                                        2777.35
                    2105                                        3341.82
                    2110                                        3946.92
                    2115                                        4606.82
                    2116                                        4751.02
                    2117                                        4900.00
ENDTABLE
WSDATA      2C  A   73          13.52       2.17
PDIRECT     1       6           11          12          28.3
POOLDATA    ELEV    2063        2063                    2030        2055        SC
PSINLET             .7          19.33
PSDATA      1       560         42                      .012        2033
ASDATA      41 1000                         1.75
ASCREST     ELEV    2094
ASINSURF    41                  6           5
ASINLET     41      0.0         0.0         30          0.0
ENDTABLE
ASINLET     41      315         2088
ASEXSURF    41                  6
                                .85
                                1
                                1
                                .001
ENDTABLE
ASEXIT      41      Y           1.0         1270        2090
BTMWIDTH    FEET    190
ASMATERIAL1
             1      15          0.001       20          105         .08
             2      0           10          0           140         3
             3      0           36          0           140         150
ENDTABLE
ASCOORD      1      TOPSOIL
            355     2088        970         2097        1000        2096
            1270    2090        1272        2089
ENDTABLE
ASCOORD      2      SHALE 1
            355     2087        970         2093        1000        2093
            1101    2091.5
ENDTABLE
ASCOORD      3      SHALE 2
            355     2085        980         2087        1101        2091.5
            1272    2089        1310        2065        1360        2055
ENDTABLE
GO,DESIGN LP                                                        2064.13
ENDJOB
ENDRUN
```

3) Case C

```
DAMS2      03/01/96  C          SAMPLE JOB C                                  C
*
*          DATA DEVELOPED TO ILLUSTRATE PROGRAM AND INTERFACE I/O
*             EXISTING AUXILIARY SPILLWAY
*          N0 ACTUAL SITE OR DESIGN IS REPRESENTED
STRUCTURE  SST1       STRUCTUREC
                      68          0.04
                      70          11.8
                      72          18.9
                      74          27.8
                      76          36.69
                      78          49.43
                      80          59.74
                      82          71.03
                      84          81.55
                      86          92.8
                      88          104.8
                      90          117.5
ENDTABLE
WSDATA     2B  A      70          2.31        3.5
PDIRECT    1.0        6           11           9.4         16
POOLDATA   ELEV       81          81                       65.7        67          SC
PSINLET                1           33
PSDATA     1          160         66                       .012        70.6
ESSPRFL    41         0.5
           50         82          65          82.5         500         84.7
           550        84.7        826         77.8         1000        77
           1130       74          1300        67
ENDTABLE
ESSURFACE  41         1000        0.01
           50         65          7.6         .5           1           5
           65         1000        5.6         .75          1           5
           1000       1300        7.6         .5           2           5
ENDTABLE
ESDATA     41                                 3
BTMWIDTH   FEET       100
ESMATERIAL1           2
            1         0           .03         -            100         .01
            2         0           .01         0            115         .02
            3         9           .002        25           100         .04
            4         0           .01         5            115         .02
            5         14          .01         18           95          .1
            6         0           .01         0            106         .02
            7         87          .0004       46           100         .14
            8         16          .001        45           85          .04
            9         0           .02         0            100         .01
ENDTABLE
ESCOORD     1         SP1
           315        94          435         91.5         475         92
           550        91.5        600         92           650         91
           735        88          750         85.5         800         83.5
ENDTABLE
ESCOORD     2         SM1
           120        85          220         88.5         275         93
           315        94          550         87
ENDTABLE
ESCOORD     3         ML
           250        87          315         91           400         87
ENDTABLE
ESCOORD     4         SM2
```

```
          600       87        800       83.5      985       78
          1000      77        1130      74        1300      67
ENDTABLE
ESCOORD    5        CL
          50        82        120       85        250       87
          315       88.2      400       87        475       86
          550       87        600       87        800       77
          1300      60
ENDTABLE
ESCOORD    6        SM3
          50        74.7      300       79        600       72
          650       71.5      900       70        1300      56
ENDTABLE
ESCOORD    7        CH
          300       79        475       80        600       72
ENDTABLE
ESCOORD    8        MH
          650       71.5      800       75        900       70
ENDTABLE
ESCOORD    9        SP2
          50        64.5      475       70        800       68
          1300      51
ENDTABLE
GO,DESIGN LCP                                                 81
ENDJOB
ENDRUN
```

4) Case F

```
DAMS2     03/01/96  F         SAMPLE JOB F                                F
*
*         DATA DEVELOPED TO ILLUSTRATE I/O
*         MODIFIED F TO USE ESSPRFL
STRUCTURE SITEF     STRUCTUREF
                    2063                                          0.0
                    2065                                          81.33
                    2070                                          315
                    2075                                          596
                    2080                                          929
                    2085                                          1322
                    2090                                          1767
                    2095                                          2252
                    2100                                          2777
                    2105                                          3341
                    2110                                          3946
                    2115                                          4606
                    2116                                          4751
                    2117                                          4900
ENDTABLE
WSDATA    2A1 A     73        13.52     2.17
STORM                         24
RAINTABLE RTA       24        PMP
                    0.0       0.2       0.5       0.8       1.1
                    1.4       1.7       2.4       3.1       3.9
                    4.7       5.5       6.3       8.9       11.8
                    14.8      29.9      32.7      35.2      35.5
                    35.8      36.1      36.4      36.7      37.0
ENDTABLE
```

```
POOLDATA  ELEV              2063              2030      2055      SC
PSINLET             0.70    19.33
PSDATA    1         560     42                .012      2033
ESSPRFL   41        0.0
          355       2088    970       2094    1000      2094
          1270      2090    1272      2089    1310      2065
          1360      2055
ENDTABLE
ESSURFACE 41        1270    .001
          355       1000    5         0.8     1         1
          1000      1270    5         .8      2         1
          1270      1360    4         .7      3         .5
ENDTABLE
ESDATA    41                          1.75
BTMWIDTH  FEET      190
ESMATERIAL1         1
           1        15      .001      -       105       .4
           2        0       10        0       140       3
           3        0       36        0       140       150
           4        10      .01       20      100       .2
ENDTABLE
ESCOORD    1        TOPSOIL
          355       2088    970       2094    1000      2094
          1270      2090    1272      2089
ENDTABLE
ESCOORD    2        SHALE 1
          355       2087    970       2093    1000      2093
          1101      2091.5
ENDTABLE
ESCOORD   3    SHALE 2
          355       2085    980       2087    1101      2091.5
          1262      2089.15 1300      2055
ENDTABLE
ESCOORD    4        SOIL
          1262      2089.15 1272      2089    1310      2065
          1360      2055
ENDTABLE
GO,STORM  L         RTA               1
ENDJOB
ENDRUN
```

2

VITA

Qi Liu

Candidate for the Degree of

Master of Science

Thesis: GUI DESIGN AND IMPLEMENTATION FOR SITES APPLICATION

Major Field: Computer Science

Biographical:

Personal Data: Born in SuZhou, P.R. Of China, On May 27, 1962, the son of Chen Wang and Dar Lin Liu.

Education: Graduated from Dongshan High School, Suzhou, Jiangsu, China in June 1979; received Bachelor of Arts degree in History from Suzhou University, Suzhou, Jiangsu, China in June 1983; received Master of Arts degree in History from East China Normal University, Shanghai, China in June 1989. Completed the requirements for the Master of Science degree at Oklahoma State University in December 1996.

Professional Membership: History Society of China.