Programmer's Guide
to the
Landfill Air Emissions Estimation Model

DOS Version 2.0
Windows Version 1.0

Prepared by

RLB Enterprises
12204 Inglehurst Drive
Raleigh, NC 27613

Table of Contents

Introduction 5
Landfill Model Overview 5
DOS Model 6
System Description Overview 6
Component Descriptions and Sources 6
Compilation Instructions 6
Executable Considerations 6
Internal Framework 7
Internal Data Structures 7
System Parameters 7
Model Parameters 8
Landfill Parameters 9
Results 10
File Formats 11
Help System Reference 11
Configuration Maintenance 11
Defaults Maintenance 12
Study Landfill Maintenance 13
Operating Description 13
Program Initialization 13
Operating Flow 15
Module Description 16
Include Files 19
Windows Model 20
System Description Overview 20
Component Descriptions and Sources 20
Compilation Instructions 20
Executable Considerations 20
Internal Framework 21
Internal Data Structures 21
Model Data 21
Results Data 27
Reports Data 28
File Formats 28
System Configuration 28
Study Information 29
Resource Information 29
Class Types 30
Windows Messages 30
Subclassed Controls 32
Program Operating Description 33
Program Initialization 33
Operating Flow 33
Interoperability 35

Module Descriptions 37
Include Files 39

Appendix A. DOS Version NMAKE File
Appendix B. DOS Version Code Listings
Appendix C. Windows Version Linker Definition File
Appendix D. Windows Version Resource File
Appendix E. Windows Version Source Code Listings

List of Figures

Figure 1 - Functional Outline 15
Figure 2 - Functional Diagram of Windows Model 34

List of Tables

Table T- 1. Help Index Marker Mappings 12
Table T- 2. Toxic Compound Defaults File Format 13
Table T- 3. Parameter File Format (.PRM) 14
Table T- 4. Interpretation of Extra Bytes in the Windows Class Structure 30
Table T- 5. Custom Windows Messages Used in the Landfill Model 31

Introduction

In the early part of 1990, Radian Corporation was commissioned by the Control Technology Center of the U.S. Environmental Protection Agency to develop an IBM PC compatible based model to assist state agencies and private landfill owners/operators in estimating gaseous emissions from landfills. In order to support the widest variety of systems, Radian developed the Landfill Air Emissions Estimation Model (LAEEM) version 1.0 as a DOS based system with minor graphics capabilities and limited projection capabilities. In order to tailor the system to the projection needs of global climate programs, the model was also immediately revised to allow for greater projection (version 1.1).

From that time, the computer industry as a whole has experienced a variety of revolutions. Current desktop computer systems are cheaper, more powerful, and more sophisticated than five years before. In particular, in the IBM compatible world, the systems moved to more graphical environments, with Microsoft Windows becoming the dominant force in the operating system marketplace.

In the venue of environmental regulation, there were similar revolutions during the same time. Most notably, the introduction of the Clean Air Act and the impending promulgation of a landfill regulation both added increased emphasis in the need to estimate emissions from landfills.

As a result, in 1994, the Control Technology Center sought to update the LAEEM to incorporate these changes in regulation and to bridge the system into newer technologies. Again, the agency contracted with Radian Corporation to revise the older model to incorporate the changes as well as develop an entirely new system for Microsoft Windows.

In an effort to preserve continuity in any future revision/development efforts associated with the LAEEM, this document provides essential systems documentation of the model, with the intended audience being future programmers.

Landfill Model Overview

The core of the Landfill Air Emissions Estimation Model is a first order exponential decay model. Specifically, the landfill emissions estimates are derived from the following equation:

$$f(t) = 2l_0 \sum_{0}^{n} M_n e^{-kn\Delta t}$$

where:

- $\mathbf{l_0}$: the methane generation capacity. This is the total amount of methane that will be able to be generated from a given amount of refuse;
- $\mathbf{k}$: the methane generation decay rate. This parameter governs the rate at which methane is released from a given amount of refuse;
- $\mathbf{M_n}$: mass of waste element. The above equation assumes waste is added as elements to the landfill and that the decay of the refuse commences immediately.
- $\mathbf{\Delta t}$: the time element size.

The presence of the constant 2 in the above equation is indicative of an assumption made regarding the landfill gas; namely, the primary constituents of landfill gas are carbon dioxide and methane and all other compounds make up a negligible percentage of the total volume. In particular, the assumption is that the landfill gas is composed of nearly equal parts of the primary constituents. Since the equation is primarily based on methane estimates, the presence of the 2 indicates inclusion of the carbon dioxide in order to estimate the total gaseous volume.

DOS Model

System Description Overview

The DOS implementation of the Landfill Air Emissions Estimation Model was programmed in C using the Microsoft C compiler (version 8 at the time of production). The user interface is text based (console I/O) with minor graphics capability for viewing the results. Beyond modifications to the system to support Clean Air Act provisions, the code is very nearly the same as the previous revisions.

Component Descriptions and Sources

Apart from the routine runtime library support provided by Microsoft C, there are two additional component libraries used to develop the system.

The entire user interface is text based and uses the CScape windowing library from Oakland Software. This version of the library is circa 1987. Oakland Software made considerable revisions to the library about 1992 which would have meant significant changes to the model code to support it. Rather than invest time in what was seen to be an obsolescing version of the model, all model recompiles where done using this older version of the library. No anomalies have been observed in using the system.

All graphical elements of the program rely on the standard Microsoft C graphics functions. The graphics display facility is used to depict the model results as an XY plot. This is accomplished using the presentation graphics component library included in the Microsoft C distribution.

Compilation Instructions

The Microsoft NMAKE makefile used to generate the executable is attached as Appendix A. The program is built assuming the large memory model with a stack size of 16000 bytes and a far segment threshold size of 256 bytes.

Executable Considerations

Upon creation, the bare executable size is over 200K with the program requiring upwards of 500K to run successfully. After creation, the program was customarily compressed using public domain LZH executable compression utilities to yield a distribution size of approximately 125K.

Internal Framework

Internal Data Structures

There are several central data structures which attempt to encapsulate all data relevant to the modeling procedure. They are classified as follows:

- System Parameters
- Model Parameters
- Landfill Parameters
- Results

System Parameters

System parameters define the overall operation of the program, independent of the landfill of interest. In the DOS model, the system parameters are fairly minimal and are defined as a structure of the following form:

```
struct system_parms {
        char working_dir[256];
        char landfill_dir[256];
        int print_device;
        int default_mode;
        int years_passed_closure;
        };
```

Working Directory (working_dir). The working directory is location where individual landfill files (studies) reside. On first execution of the program, the working directory is taken from the command line as being the directory containing the landfill executable. This is saved in the DOS model configuration file for future use. This parameter is user editable via the user interface.

Landfill Directory (landfill_dir). The landfill directory is the location of the program and support files. On execution, the landfill directory is obtained from the command line. This allows the user to execute the model from a directory in the search path and the model still being access to configuration, help, and data files.

Print Device (print_device). The print device specifies DOS I/O device (e.g., LPT1) to which printed text reports are sent. This parameter is user selectable.

Default Mode (default_mode). The default mode specifies the data source used to initialize model parameters. Currently, there are two such modes: Regulatory (CAA) and AP42. When a new study is created, model parameters is intialized with values recommended from the appropriate data source. This parameter is user selectable.

Years Passed Closure (years_passed_closure). This parameter specifies the number of years passed the landfill closure date (the year after which the landfill will no longer accept waste). This is a user selectable parameter and defaults to 50.

Model Parameters

The Model Parameters structure house the working defaults for the program:

```
struct model_parms {
        double l0, k, nmoc, ch4, co2;
        int include_air_toxics;
        struct air_toxic_structure *air_toxics;
        int number_air_toxics;
        };
```

l_0 (l0). The methane generation capacity in Mg/m^3.

k. The methane generation decay rate constant in yr^{-1}.

NMOC Concentration (nmoc). The concentration of nonmethane organic compounds in landfill gas.

Methane Percentage (ch4). The percentage of landfill gaseous emissions made up of methane.

Carbon Dioxide Percentage (co2). The percentage of landfill gaseous emissions made up of carbon dioxide.

Include Air Toxic Calculations Flag (include_air_toxics). A flag to indicate that air toxic emissions should be calculated and tracked.

Toxic Emissions Information (air_toxics). A pointer to the location of the air toxic data (structure described below).

Toxic Emission Compound Count (number_air_toxics). A count of the number of toxic compounds in the air toxic data structure.

The model parameters structure contains pointer to an array of air toxic compounds (up to *number_air_toxics*). An individual array element houses information in the following structure:

```
struct air_toxic_structure {
        char chemical_name[50];
        double molecular_wt;
        double concentration[2];
        };
```

Chemical Name (chemical_name). The name of the compound.

Molecular Weight (molecular_wt). The molecular weight of the compound.

No Codisposal/Codisposal Concentration (concentration[2]). Individual compound emissions are calculated by multiplying the calculated gaseous emissions by statistically derived factors, stratified by whether the landfill has historically collected hazardous waste (a codisposal site). This element is an array of two doubles for the no codisposal and codisposal cases, respectively.

Landfill Parameters

With respect to the model, there are several parameters which define a landfill. They are organized as follows:

```
struct landfill_parms {
        int year_opened, current_year, year_closed;
        double capacity, avg_acceptance_rate, last_rate;
        int number_of_periods;
        double *refuse_in_place;
        };
```

Year Opened (year_opened). The year the landfill began accepting waste.

Current Year (current_year). The last year for which acceptance rate data is available. This need not be the actual current year.

Year Closed (year_closed). The last year waste is expected to be accepted.

Design Capacity (capacity). The maximum amount of waste expected to be landfilled (or the permitted capacity).

Average Acceptance Rate (avg_acceptance_rate). The average rate waste is added to the landfill.

Last Acceptance Rate (last_rate). The last non zero acceptance rate.

Number of Yearly Periods (number_of_periods). The number of yearly refuse values the program will allow to be entered (equal to current_year - year_opened + 1).

Refuse in Place (refuse_in_place). A dynamic array of doubles (with number_of_periods elements) containing refuse totals (in Mg) for a landfill.

The Landfill Air Emission Estimation Model also provides for the calculation of the methane decay rate constant using the EPA Method 2E. Where applicable, these parameters are also maintained in the following structure:

```
struct k_calc_parms {
        double well_depth, radius, density;
        double fraction, flow_rate, refuse_age;
        };
```

where the parameters are:

Well Depth (well_depth). The depth of a landfill gas extraction well, in feet.

Well Radius of Influence (radius). The radius around the well from which gas is expected to be removed, in feet.

Refuse Density (density). The density of the landfill waste, in Mg/ft^3.

Fraction of Decomposable Refuse (fraction). The fraction of the landfill waste that is expected to decompose and produce landfill gas.

Well Flow Rate (flow_rate). The rate at which gas is extracted, in ft^3/min.

Average Age of Refuse (refuse_age). The average age of the refuse within the radius of influence of the well, in years.

Results

After the user has entered all the required data, the model can be asked to calculate emissions. Once a calculation is done, all the resulting values are maintained until another calculation is initiated by the user. In this way, the results from which reports are generated can be independent of the current state of the default parameters.

All results are maintained in a memory structure of the following form:

```
struct model_results {
        int year_count;
        int air_toxics_count;
        int *year_list;
        double *refuse_data;
        double *annual_ch4;
        double *annual_co2;
        double *annual_nmoc;
        double **annual_air_toxics;
        };
```

where the component values are:

Count of years (year_count). The number of year values, and, therefore, the number of array elements, for each of the results elements contained in the structure.

Count of air toxic compounds (air_toxics_count). The number of air toxic compounds for which results maintained in the structure.

List of Years (year_list). This list of years for which the data applies.

Refuse Data (refuse_data). An array of the yearly refuse values for the landfill.

Annual Methane (annual_ch4). An array of the annual methane values for the landfill.

Annual Carbon Dioxide (annual_co2). An array of the annual carbon dioxide values for the landfill.

Annual Nonmethane Organic Compound Emissions (annual_nmoc). An array of the annual NMOC emissions for the landfill.

Annual Emissions from Tracked Air Toxic Compounds (annual_air_toxics). An array of arrays of annual emissions from each of the tracked air toxic compounds. This number of compounds in this array is *air_toxics_count*, each with *year_count* elements.

File Formats

The model uses external files for a variety of purposes. The major uses for external files are:

- Help System Reference
- Configuration Maintenance
- Defaults Maintenance
- Landfill Study Maintenance

Each use and its file format is described below.

Help System Reference

The help system provided by the model draws all information displayed from an external help file called LANDFILL.HLP. This file is an ASCII format file with index marks above the major headings for each help topic. This allows for easy revision of the help text.

An index mark in the help file is indicated by a "@@" ASCII sequence followed by a number indicating the section. The program assumes all text that follows the "@@#" marker is help text for the current help selection. The program ceases reading text when the system encounters another index marker or the end of file.

The input system does not strip the leading bit from the input characters so the system will display OEM specific characters. Much of the current help file was annotated with markers that use line draw characters in the IBM PC OEM character set to mark headings in the help facility.

Table T-1 describes the index marker mappings for the help system:

Configuration Maintenance

Although the model does not require many pre-configured parameters, there are a few that are maintained between executions of the model in a file called LANDFILL.CFG. If this file is missing on startup, it is regenerated from the default settings and saved in the landfill program directory.

The configuration file is a plain ASCII file with four configuration settings, each delimited by a newline. The maintained parameters (in order of occurrence) are:

- *Working Directory (char 255)*: a full path specification to the directory containing the study data. This directory is set in the CONFIGURE PROGRAM dialog.
- *Print Device (integer)*: logical printer output device where 0 is LPT1, 1 is LPT2, etc.
- *Default Mode (integer)*: default values used to select default values used in creating a new study where 0 is regulatory and 1 is AP42.
- *Years Passed Closure (integer)*: number of years beyond the expected landfill closure data for which the system will estimate emissions.

Marker Number	Description
@@1	Main Menu Help
@@2	Study Type Menu Help
@@3	Study Name Dialog Help
@@4	Edit Study Data Menu Help
@@5	Edit Chemical Composition Dialog Help
@@6	Edit Air Toxic Composition Dialog Help
@@7	Edit Methane Generation Rate Constant Menu Help
@@8	Edit Methane Generation Rate Constant Dialog Help
@@9	Using the Calculate Methane Gas Generation Constant Dialog Help
@@10	Edit Methane Generation Capacity Dialog Help
@@11	Edit Refuse Data Periods Entry Dialog Help
@@12	Display results Menu Help
@@13	Tabular Report Menu Help
@@15	Report Selector Help
@@16	View Model Results Help
@@17	Edit System Configuration Dialog Help
@@18	Save Menu Help
@@19	Output Filename Dialog Help
@@20	Closure Periods Dialog Help
@@21	Select File Dialog Help

Table T- 1. Help Index Marker Mappings

Defaults Maintenance

Under the specifications for the first landfill model, the system was to provide a series of default data where possible to simplify the process of estimation of emissions. For the most part, the data for parameters for which defaults could be provided was limited to a single value. Consequently, the majority of the default values were maintained in source code and statically linked into the executable. These constants are propagated through the code via C preprocessor defines and are maintained in DEFAULTS.H.

Air toxic emissions data, however, were the exception to the rule. The original specification for the air toxic compound list was such that it was dynamic and could be revised as new data was received. This implied the list of compounds could grow, shrink, or be modified in any way and provisions should be made to allow revision of the list apart from creating a separate executable. The only facility available to provide this capability was storing the information in an external, plain text file. In the current implementation of the system, this file is called AIRTOXIC.TXT.

In the Air Toxic defaults file, each compound is an ASCII record delimited with a newline. The format of each record is shown in table T-2.

Field	Type	Width
Compound Name	char	28
Molecular Weight	double	free
Concentration for No Codisposal Landfills	double	free
Concentration for Codisposal Landfills	double	free

Table T- 2. Toxic Compound Defaults File Format

Study Landfill Maintenance

The focus of the landfill model is the given conditions and operation of a specific landfill. For a given landfill, the user is required to enter yearly refuse quantities and may optionally modify default parameters. All of data entered may be saved for future modeling and reference as an ASCII file with a user specified name. In practice, the process of saving a study actually saves two separate files with different extensions, PRM and ATX.

The ATX file is the air toxic compounds file with any user modifications. This file follows the same format as the default air toxic compound file.

The PRM file is the parameter file, which encapsulates all of the working parameters relating to the landfill. These parameters include the model equation parameters, operating periods of the landfill, and the refuse quantities by year. The format for this file is shown in Table T-3.

Operating Description

The LANDFILL executable is invoked as a single command without parameters. The system performs some minor initialization steps and presents the user with a text menu showing the available functionality.

Program Initialization

The entry point for the program is contained in the source module *main.c*. This module houses the control structures which define the overall program flow as well as routines that relate to study loading/saving and initialization.

When the program is loaded, the system performs the following initialization steps:

- Initializes the screen and sets the color scheme for the program;
- Performs a memory check to determine if the available memory is sufficient for running the program;
- Displays an introduction screen;
- Sets the default system configuration either using a previously created configuration file or using system generated defaults and saving them to a new configuration file; and,
- Enters the main system loop.

The memory checking procedure is the most esoteric feature of the initialization step. DOS memory management services are slow and most compilers provide alternative heap management

procedures in order to speed the memory allocation process. This is usually done by providing program initialization code within all executables that allocate a predetermined large block of system memory and satisfying all program allocation requests by partitioning the blocks into smaller elements. Requests that exceed the initial block size are satisfied by a subsequent call to the DOS memory allocation service.

Most memory management functions provided by runtime libraries only deal with heap sizing. Also, at the time, functions were not available to size the remaining pool of unallocated DOS memory. To circumvent this limitation, a feature of the DOS allocation service was used to size the total memory.

When an allocation request is made of DOS, there are two possible outcomes: success and failure. In the case of success, the system returns a handle to the memory block. In the event of failure, a code is returned indicating failure along with the minimum allocation request size that will result in success. The landfill model system uses this fact in that it initially requests DOS allocate all system memory to the model. Since this will fail, the model examines the value returned from DOS and, by adding the available heap size to return quantity, a good approximation of available system memory was obtained. This is compared against a stored value to determine if the memory is sufficient to run the program.

Field	Type	Width
Record 1		
Methane Generation Capacity (l_0)	double	All parameters are free
Methane Generation Decay Rate (k)	double	
Non Methane Organic Compound Concentration	double	
Percentage Methane in Landfill Gas	double	
Percentage Carbon Dioxide in Landfill Gas	double	
Flag for Including Air Toxic Compounds	bool (1/0)	
Record 2		
Year Landfill Opened	integer	
Current Landfill Year	integer	
Year Landfill Closed	integer	
Record 3		
Landfill Capacity	double	
Average Refuse Acceptance Rate	double	
Last Nonzero Refuse Acceptance Rate	double	
Record 4		
Flag for Data Completion Permitting Calculation	bool (1/0)	
Record 5		
Number of Yearly Refuse Quantity to Follow	integer	
Record 6-n (where n is quantity in Record 5)		
Refuse Quantity	double	

Table T- 3. Parameter File Format (.PRM)

Operating Flow

The LANDFILL executable is invoked as a single command without parameters. The system performs some minor initialization steps and presents the user with a text menu showing the available functionality. Figure 1 provides a outline of this functionality.

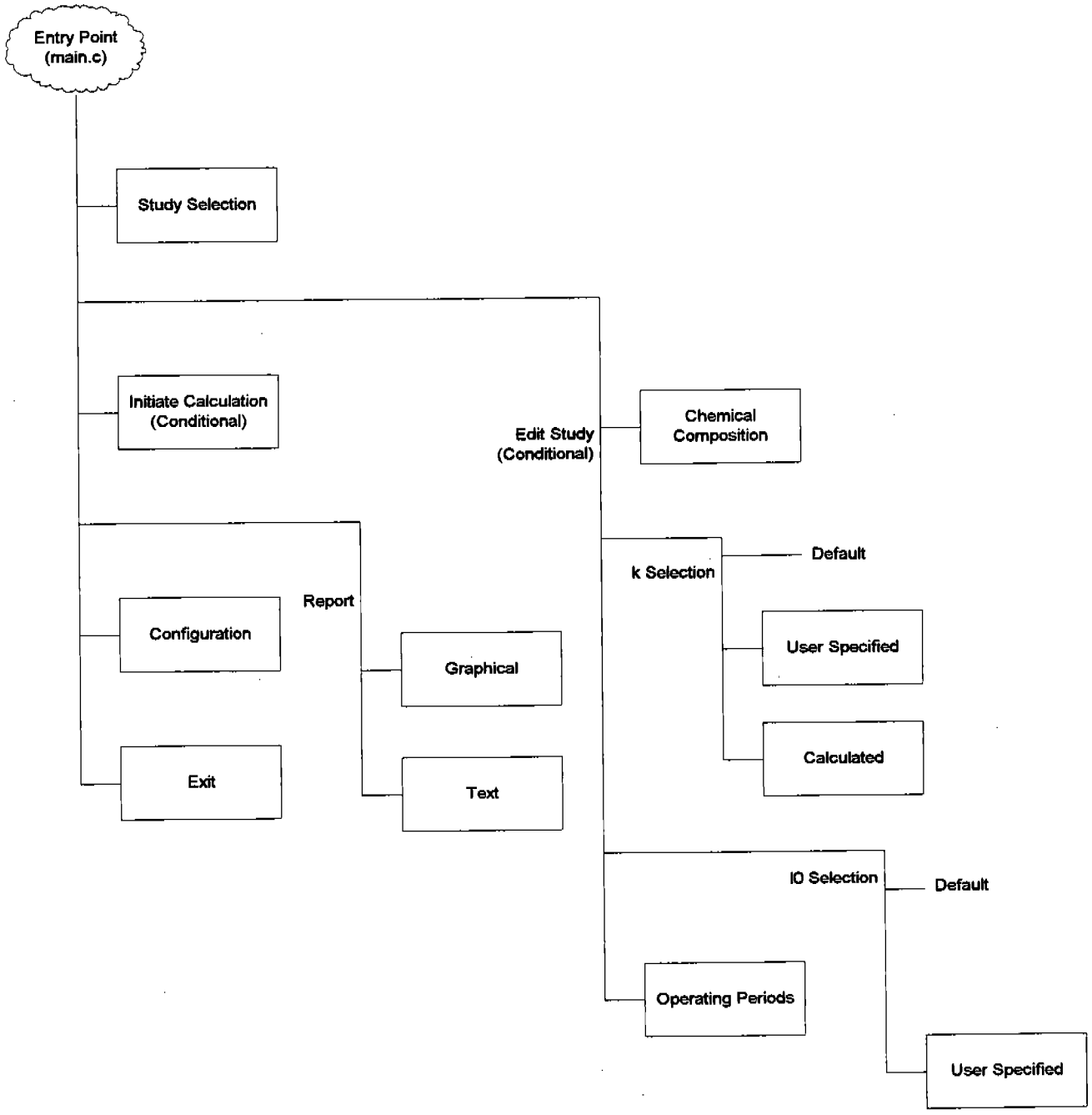


Figure 1 - Functional Outline

Module Description

Appendix B contains code listings for all modules used in the system. This section provides a succinct description of each module and contents. The source code listings provided contain documentation which expands on these discussions.

Main.c & Mainmenu.c

As well as providing initialization and the functional control structure depicted above, the module main.c contains routines for loading or initializing landfill studies. Mainmenu.c contains the menu code.

New-old.c

This module controls the display of the new/old study selection menu.

Studname.c

This module the dialog box for receiving the name of a study. The edit field in this dialog box is an revised control which filters entry for only DOS characters which may be used in a filename (*FNDOSFIL.C*).

Selector.c

This module houses the menu code for displaying the names of landfill studies in the working directory.

Findfls.c

This module houses a routine to get the names of the studies in the working directory and return to the caller.

Editmenu.c & Editmain.c

These two source modules work together to get and process user input relating to the entire edit hierarchy of the functional description. Editmenu.c manipulates the display facility to display the menu of chioices and Editmain.c handles the outcome.

Chemcomp.c

This source module provides for entry of the chemical composition information: percentage of methane and carbon dioxide in gas, concentration of NMOC, and the desire to include air toxic information.

Airtoxmn.c

This source module provides for the entry and maintenance of the individual air toxic compounds.

Methrate.c

This source module displays and handles selection of the k selection menu.

Kedit.c

This source module handles the k entry dialog box.

Kcalcmen.c

This source module handles the dialog box which controls the method 2E calculation procedure.

Kcalcmn.c

This source module contains the code necessary to calculate a k value. The k calculation procedure is basically a root finding method based on the bisection and Newton-Raphson numerical methods.

The equation from which the k is determined has up to two roots. This algorithm will automatically restrict the solution to the root closest to the y axis by initially using bisection to narrow the iteration interval as close as possible to the root which lies between 0 and the derivative of the k function. After a preset number of bisection iterations, Newton-Raphson is used to further refine the root.

Methrate.c

This source module handles dialog box used to enter/revise the l_0 value.

Periods.c

This module controls the dialog box for entering the landfill operating parameters.

Closure.c

This module controls the display of the closure year dialog box information screen. Depending on the parameters entered, this box will allow editing of the displayed closure year.

Calcmain.c

This module is the heart of the model. The code within this file handles all the calculations.

Disprslt.c

This module displays a summary of the calculation after the model is finished running a scenario.

Do_graph.c

This module formats and displays the results in a graphical format.

Tblemenu.c

This module displays a menu allowing selection of the destination of a text report.

Strmview.c

This module generates the text report and is used to output the report to either the printer or to a file.

Scr_view.c

This module generates and displays the text report on the screen.

Config.c

This module handles the entry/selection of model configuration parameters.

Savemenu.c

This module displays a the save study selection menu.

Windows Model

System Description Overview

The Windows implementation of the Landfill Air Emissions Estimation Model was developed with the DOS model as a backdrop and, therefore, shares some common elements in relation to organization and functionality. However, a simple port of the system to the new environment would have ignored key features of the Windows operating system. Given the assumption that Windows is in the future of IBM compatible computing, a reasonable implementation of the model should take advantage of the significant features of the environment.

The Windows implementation of the Landfill model attempts to embrace those features and, as a result, the system is functionally extended. The Windows model has the following characteristics over the DOS implementation:

- Multiple Document Interface (MDI) based which allows for multiple landfill scenarios to be edited simultaneously;
- Takes advantage of multitasking capabilities by providing simultaneously accessible tools and on revision recalculation;
- Printing/display capabilities;
- Enhanced memory usage;
- Better defaults value handling through resource files.

Although the system was developed concurrently with the promulgation of the Win32 standards, the decision was made to develop entirely within the Win16 environment without special provisions for 32 bit systems. This was specifically due to some uncertainty in the Win32 compatibility code.

The system was developed using the Borland C++ compiler (version 5.0 at the time of last production).

Component Descriptions and Sources

No other library components were used apart from the routine runtime library support provided by Borland C++. The system does use some elements of the Borland custom control library.

Compilation Instructions

The model was developed entirely with the Borland C++ Development Environment and handled entirely within the interface. The system was developed using the large memory model with a stack size of 32767 and a local heap size of 16384. Appendix C depicts the linker definition file used to generate the system.

Executable Considerations

Upon creation, the executable size is approximately 170K and requires the presence of the BWCC.DLL file in the working or system directories. Although it is customary to ship the system with a unit converter application, these other files are not required for the system to function.

The system was specifically targeted toward Win16 environments. In particular, some Windows messages and function calls use syntax which differs from Win32 counterparts. Further, the system itself is fundamentally incompatible with Windows NT. However, the system performs quite well in the Windows 95 and Windows 3.1X environments.

Internal Framework

Internal Data Structures

Model Data

Because of the MDI nature of the Windows version, the individual data elements must be managed independently for each instance of a landfill study. Further, additional information must be track in order to handle the context of each landfill. Therefore, the data element organization for a given landfill has to be centralized in nature for efficiency.

In the model, the central data elements is the OPERDATASTRUCT, which is defined as follows:

```
typedef struct  {
        char Filename[MAXPATH];
        MODELPARAMETERS Model;
        METHOD2E Method2E;
        OPPARAMETERS Parameters;
        PARAMETERINDEX Choices;
        PERIODLIST Periods;
        AIRTOXICLIST AirToxics;
        int CellIndex, CellOffset;
        int CellCount;
        int Mark;
        } OPERDATASTRUCT;
```

where the constituents are

Study Filename (Filename). The file into which the study data is saved.

Model Parameters (Model). Data which defines the model behavior.

Method 2E Parameters (Method2E). Data relevant to the generation of k values using method EPA Method 2E testing.

Operating Parameters (Parameters). Data which characterizes the operation of a landfill.

Selection Parameters (Choices). Data which determines the selections depicted on the menu.

Operating Period Information (Periods). Data on the year to year operating characteristics of the landfill.

Air Toxic Compound Information (AirToxics). Data that defines the compounds that are tracked by the model.

CellIndex, CellOffset, CellCount, Mark. Variables which maintain state information for the display/edit facility. The *CellIndex* and *CellOffset* maintain position information for use in

determining the displayed values on the data editing screen. The *CellCount* represents the number of displayed values on the operating parameters windows. The *mark* is used to maintain the beginning of a drag selection.

Model Parameters

The model parameters are encapsulated in the following structure:

```
typedef struct {
        double l0, k;
        double PercentCH4, PercentCO2;
        double NMOC, MoleWt;
        double Temp, Pressure;
        } MODELPARAMETERS;
```

where

l_0 (l0). The methane generation capacity in Mg/m^3.

k. The methane generation decay rate constant in yr^{-1}.

Methane Percentage (PercentCH4). The percentage of landfill gaseous emissions made up of methane.

Carbon Dioxide Percentage (PercentCO2). The percentage of landfill gaseous emissions made up of carbon dioxide.

NMOC Concentration (nmoc). The concentration of nonmethane organic compounds in landfill gas.

Molecular Weight of NMOC (MoleWt). This is the assumed molecular weight of NMOC (identical to Hexane in the current implementation). This value is loaded from the executable resource information and is not user editable.

Ambient Temperature (Temp). Temperature at which emissions are generated. This value is loaded from the executable resource information and is not user editable.

Ambient Pressure (Pressure). The pressure at which emissions are generated. This value is loaded from the executable resource information and is not user editable.

Air Toxic Parameters

The air toxic compound information is tracked in the following structures:

```
typedef struct {
        int AirToxicCount;
        HGLOBAL hAirToxic;
        } AIRTOXICLIST;
```

where

Number of Compounds (AirToxicCount). The number of air toxic compounds.

Compound Data Handle (hAirToxic). The global memory descriptor for the memory which contains the air toxic information.

The handle *hAirToxic* points to an area of global memory that is *AirToxicCount* in length and has the following structure:

```
typedef struct {
        char Name[50];
        double MoleWt;
        double EmFact[2];
        } AIRTOXIC;
```

where

Compound Name (Name). The name of the toxic compound.

Compound Molecular Weight (MoleWt). Molecular weight of the compound.

Emission Factors for Compound (EmFact). An array of two emission factors for codisposal and no codisposal landfills, respectively, applied to the total amount of landfill gas that reflect the emissions of the compound.

Method 2E Parameters

The method 2E parameters are tracked in the following structure:

```
typedef struct {
        double WellDepth, ROI, Density;
        double Fraction, FlowRate, RefuseAge;
        } METHOD2E;
```

where

Well Depth (WellDepth). The depth of a landfill gas extraction well, in feet.

Well Radius of Influence (ROI). The radius around the well from which gas is expected to be removed, in feet.

Refuse Density (Density). The density of the landfill waste, in Mg/ft^3.

Fraction of Decomposable Refuse (Fraction). The fraction of the landfill waste that is expected to decompose and produce landfill gas.

Well Flow Rate (FlowRate). The rate at which gas is extracted, in ft^3/min.

Average Age of Refuse (RefuseAge). The average age of the refuse in within the radius of influence of the well, in years.

Operating Parameters

The landfill operating parameters are tracked in the following structures:

```
typedef struct {
        int YearOpened, YearCurrent, YearClosed;
        double Capacity;
        double AcceptRate, LastAcceptRate;
        PERIODLIST OpRecord;
        } OPPARAMETERS;
```

where

Year Opened (YearOpened). The year the landfill began accepting waste.

Current Year (YearCurrent). The last year for which acceptance rate data is available. This need not be the actual current year.

Year Closed (YearClosed). The last year waste is expected to be accepted.

Design Capacity (Capacity). The maximum amount of waste expected to be landfilled (or the permitted capacity).

Average Acceptance Rate (AcceptanceRate). The average rate waste is added to the landfill.

Last Acceptance Rate (LastAcceptRate). The last non zero acceptance rate.

The final element, *OpRecord,* is not actively used. It is listed above for completeness.

Operating Periods Data

The data for the refuse acceptance data for the landfill is maintained in the following structure

```
typedef struct {
        int PeriodCount;
        HGLOBAL hPeriod;
        } PERIODLIST;
```

where

Number of Operating Years (PeriodCount). The number of refuse values recorded for the landfill.

Period Data Handle (hPeriod). A global memory descriptor for the memory containing the array of double values that represents the refuse acceptance data for the landfill.

Graphics Data

The landfill model has the ability to display results of calculations graphically. This is performed using a customized graphics subsystem which develops and tracks information needed to continually display the data.

The graphical display is contains the following components:

- A title with major and minor constituents;
- Two axes with major and minor tick marks and labels;
- An arbitrary number of data series.

The graphics facility is used for window display, printing, and for saving the graphic as a Windows Metafile (WMF). To simplify the process of rendering the graphic onto each different device, the graphics manager actually works on the basis of an independent scale based on normalized display coordinates (ndc). These coordinates are based on scaling the X and Y dimensions of the output to a linear scale between 0 and 1.

These constituents are governed by the data in the following structures:

```
typedef struct {
        TITLESTRUCT titles;
        AXISSTRUCT xAxis, yAxis;
        int seriesCount;
        HGLOBAL hSeries;
        } GRAPHSTRUCT;
```

where

seriesCount. The number of data series to be graphed.

hSeries. A global memory descriptor of the memory containing each series in a *SERIESSTRUCT* format.

TITLESTRUCT

The title governing data is maintained in the following structure:

```
typedef struct {
        char fontName[80];
        char primaryTitle[80], secondaryTitle[80];
        } TITLESTRUCT;
```

where the constituents

Graph Label Font (fontName). The name of the font used to display the titles and axes labels.

Graph Titles (primaryTitle, secondaryTitle). The major/minor titles used to label the entire graph.

AXISSTRUCT

The axes governing data is organized in the following structure:

```
typedef struct {
        double axisStart, axisEnd;
        double min, max;
        double minorInterval;
        int majorInterval;
        char axisTitle[80];
        char labelFormat[30];
        } AXISSTRUCT;
```

axisStart, axisEnd. The beginning and end of the axis on the output display in ndc.

min, max. The beginning and end of the axis in real coordinates.

minorInterval. The real coordinate interval of the minor tick marks.

majorInterval. The real coordinate interval of the major tick marks.

axisTitle. The title of the axis.

labelFormat. The format for the axis tick labels as a C printf format specifier.

SERIESSTRUCT

Each of the data series to be graphed are encapsulated in the following structure:

```
typedef struct {
        int style;
        int lineStyle;
        COLORREF lineColor, pointColor;
        int pointCount;
        HGLOBAL hPoints;
        } SERIESSTRUCT;
```

where

style. A style flag (currently no usage).

lineStyle. The style of the line as a valid Windows pen style.

lineColor, pointColor. The color of the line and point markers as a valid Windows color.

pointCount. The number of points in the hPoints list.

hPoints. A global memory descriptor of the memory containing the data in a *POINTSTRUCT* structure.

POINTSTRUCT

All of the points that are plotted are arranged in the following structure:

```
typedef struct {
        double x, y;
        } POINTSTRUCT;
```

Results Data

When the user requests the display of emissions as a report or as a graphic or revises data that affects model output, the system automatically initiates the validation and calculation process for the selected pollutant. The results generated from this process are stored in an allocated area of memory and tagged to the window created to satisfy the user's request. The results are stored in a format defined by the following:

```
typedef struct {
        int YearStart;
        int YearClosed;
        int YearCount;
        double AcceptanceRate;
        int AirToxicType;
        char Name[50];
        HGLOBAL hRate;
        } RESULTS;
```

where

YearStart, YearClosed. The beginning and end year of the operation data history.

YearCount. The number of years represented by the results.

AcceptanceRate. The rate at which data will have to be accepted to in order to achieve closure by the YearClosed.

AirToxicType. An integer indicating the emissions data represented by the results (0 is methane, 1 is carbon dioxide, 2 is NMOC, and greater than 3 represents an air toxic compound whose ordinal position on the air toxic list is n-3).

Name. The name of the compound.

hRate. A global memory descriptor of the memory array of doubles in which the results reside.

Reports Data

Although much simpler than the graphics display data structure, the text report MDI window requires an underlying data structure which is tagged to the output window. The report structure is defined as:

```
typedef struct {
        int LineCount;
        HGLOBAL hReportText;
        } REPORTSTRUCT;
```

where

LineCount. The number of lines of text in the report.

hReportText. A global memory descriptor for the area of memory containing the physical text report. This is set by the return value of *BuildReport (BuildRpt.c)*.

File Formats

As in the DOS model, the Windows version uses disk files to store/retrieve configuration and landfill study information. However, due to facilities provided by Windows, much of the process is entirely different.

System Configuration

As is customary with Windows based applications, the Windows version of the model stores configuration information in a initialization file (INI file). In particular, because the number of saved parameters is small, the Windows version of the model enters information into the WIN.INI file user a section name of [LAEEM]. An example of such an entry is as follows:

```
[LAEEM]
DefaultsMode=Regulatory
LandfillType=NoCodisposal
MaxYears=200
User_l0=169.901
User_k=0.05
User_NMOC=4000
```

The initialization strings are defined as follows:

DefaultsMode. The defaults mode specifies the source of the default parameters in the model. There are three such sources: regulatory, AP42, and user. Regulatory implies values defined for use by the Clean Air Act will be assumed. AP42 implies values referenced in AP42 documentation will be used. User implies user supplied values will be used.

LandfillType. For the purposes of the model, there are two different classifications of landfill types that effect the estimates of emissions. *Codisposal* sites are those landfills that have accepted hazardous waste now or at any time in the past. Conversely, *no codisposal* sites are those landfills that have never accepted hazardous waste. This parameter selects the default mode to use in estimating emissions.

MaxYears. The model has the capability of estimating emissions for an arbitrary period after the last waste is accepted . This parameter defines the number of years past the closure year for which emissions will be calculated.

User_l0, User_k, User_NMOC. Under the user defaults mode, the user has the ability to revise calculation governing data. These data are stored for future use under these variables.

Study Information

As with the DOS model, the Windows model is capable of storing landfill study information into a disk file. This file is functionally equivalent but fundamentally incompatible with the DOS version. It is generated by a series of Windows _lwrite statements using pointers to the C structures themselves and is compiler and system specific.

Understanding this, the C data items are written in the follow order:

- MODELPARAMETERS
- METHOD2E
- OPPARAMETERS
- PARAMETERINDEX
- PeriodCount
- Periods
- AirToxicCount
- AirToxic

Resource Information

Appendix D provides the resource listings for the resource table in the model. The defaults data for the model are contained as STRINGTABLE entries in the resource definition. Two different sections of defaults are included: general defaults and individual air toxic compounds.

General Defaults

For the most part, general defaults are easily interpreted. Of particular note is the fact that the values are entered as strings. The process that reads the resource table (located in *LOADDEF.C*) provides the conversions.

Individual Air Toxic Compounds

The STRINGTABLE section containing the air toxic compound list begins at a predefined resource identifier (defined in *LANDRES.H*) and continues sequentially until a null record is reached. An example record is as follows:

21, "1,1,2,2-Tetrachloroethane;167.85;1.11;1.11"

The first number is the resource identifier and the definition is in quotations. Within the quotation, there are four fields, delimited by semicolons. The field definitions are: (1) compound name, (2) molecular weight, (3) no codisposal emission factor, and (4) codisposal emission factor.

The last resource record should be the null record, which has the following format:

"END;0;0;0"

Class Types

As mentioned earlier, the Landfill model is an MDI application. Under the MDI framework, each distinct window type must have a distinct window class that allows Windows to route the messages to the appropriate message handler for the class. Within the Landfill model, there are three distinct types of windows that can serve as clients of the MDI frame:

- *OPERWINDOW*. This window handles the display and entry of operating data for a particular landfill and resembles a grid.
- *TEXTWINDOW*. This window handles the display and movement of the text report of calculations.
- *GRAPHWINDOW*. This window handles the display of the graphical results for a particular study.

For a given landfill, the different classes of MDI windows are linked to permit different views of the same data (i.e., the data may be edited in a window of OPERWINDOW class while the emissions estimates are simultaneously viewed using windows of TEXTWINDOW or GRAPHWINDOW class). This is accomplished by using *cbWndExtra* component of the window class structure to reserve additional structure space to maintain this link. These values are accessed using the *SetWindowLong* and *GetWindowLong* functions.

In actuality, three additional longs are reserved in the window class structure for each of these MDI window classes. The interpretation of the additional elements is shown in Table T-4.

Offset (bytes)	Interpretation
0	Global handle of the OPERDATASTRUCT used to generate the results associated with this window.
4	*OPERWINDOW*: No application. *GRAPHWINDOW*: Global handle of the GRAPHSTRUCT used to maintain the graphic display. *TEXTWINDOW*: Global handle of the REPORTSTRUCT used to maintain the text display.
8	Global handle of the RESULTS structure that maintains the current calculation results used to generate the data.

Table T- 4. Interpretation of Extra Bytes in the Windows Class Structure

The unifying feature of all the MDI client windows is the OPERDATASTRUCT global handle maintained at offset 0. This parameter is used to relate all of the windows back to their source and to permit updating report/display windows when the operational data changes.

Windows Messages

Under Windows, independent windows exist as independent entities. However, at times, windows must work in concert to accomplish a task. Under the Landfill model, this is largely accomplished using customized application messages. Table T-5 documents the custom messages used in the model.

Message	Definition (relative to WM_USER)	Usage
AM_WINDOWACTIVE	101	A message sent from the MDI client window to the MDI parent to indicate it has been activated by the user.
AM_GETACTIVETOOL	102	A request sent to the MDI parent to determine the active toolbar.
AM_MDISAVESTATE	103	A request sent to the OPERWINDOW MDI client to cause it to save the current state of the operational data from the active toolbar
AM_RESTORESTATE	104	A request sent to the OPERWINDOW MDI client to cause it to restore the operational data to the active toolbar.
AM_GETCURRENTENTY	105	A request sent to the OPERWINDOW MDI client from the active toolbar to determine the currently selected operating period.
AM_ENTRYSELECTED	106	A message sent from the OPERWINDOW MDI client to the active toolbar to inform that a new operating period has been selected.
AM_PERIODCHANGED	107	A message sent from the active toolbar to the OPERWINDOW MDI client to inform it that a new operating period was selected.
AM_CONFIRMDATA	108	A message sent to a window to request that data values in fields should be validated and the data structures updated.
AM_VALUESCHANGED	109	A message sent to an MDI client to indicate data was changed that may create the need to redisplay. This message is also used to initialize data in report windows.
AM_GETOPERHANDLE	110	A request sent to an MDI client to get the OPERDATASTRUCT global memory descriptor attached to the window.
AM_UPDATEFIELDS	111	A message sent to the active toolbar to cause it to refresh data in the tool.
AM_FLUSHFIELDS	112	A message sent to the active toolbar to cause it to update in memory structures.

Table T- 5. Custom Windows Messages Used in the Landfill Model

Message	Definition (relative to WM_USER)	Usage
AM_SAVE	113	A message sent to a report MDI client to cause it to save the information.
AM_PRINT	114	A message sent to a report based MDI client to cause it to print the information.
AM_COPY	115	A message sent to the OPERWINDOW MDI client to copy selected data.
AM_CUT	116	A message sent to the OPERWINDOW MDI client to cut selected data.
AM_PASTE	117	A message sent to the OPERWINDOW MDI client to paste selected data.

Table T-5 (cont.). Custom Windows Messages Used in the Landfill Model

Subclassed Controls

To improve the appearance and operability of the model, two of the standard windows objects were subclassed to create new types.

In the operating parameters MDI client, all of the information is physically entered and revised using a toolbar with several edit fields for integer based and floating point based data. However, these fields were intended to operate cooperatively with the OPERWINDOW MDI client.

Because the data entry focus needed to reside with these dialog items, certain keystrokes had to be handled differently from the normal default behavior in order to allow some simultaneous control of the operational data MDI client. Subclassing the default operation was the only way to accomplish this.

In order the manage the subclassing, a series of routines were created to manage the process. These routines are located in the module SUBCLASS.C. In order to preserve the linkage between the parent and child classes, the original message processing function was obtained from the class using the *GetWindowLong* function and was stored as a property using the *SetProp*. The new message handler was inserted in the place of the original message handler. The new message handler routine then has the responsibility of responding to the message, discarding the message, or passing the message to the original class definition handler. A generic edit field message handler set is provided in the SUBCLASS.C module and is used throughout the model.

For the purposes of the toolbar, the custom message handler was located inside the opertool.c module. This was necessary in order to remap the behavior of the edit controls in order to pass the UP and DOWN arrow keystrokes from the control to the OPERWINDOW MDI client and allow for positioning. Further, the behavior of the RETURN key was modified to allow the field to be validated and updated upon return while remaining of the field. For cosmetic reasons, bevels were added to the edit field.

Another custom control was created to provide the background of the toolbar and to house the column labels for the operating periods window. This module is housed in LANDCUST.C. This is basically a window with bevels which stretches the width of the parent window. Note that the client area of the MDI frame window was modified to allow for static placement.

Program Operating Description

The LANDFILL executable is invoked as a single command without parameters. The system performs some minor initialization steps and presents a limited menu showing available selections. The menu context changes after loading a landfill study.

Program Initialization

Initialization begins with the creation of the window class types to be used in the program. The LANDWIN class type application class window is created. The default values are then loaded from the initialization file (if any are present). They are then saved to the initialization file, thereby creating entries if they did not previously exist. Accelerator keys are loaded from the resource file and the system enters the main message loop.

From the time the application class window is created, an independent initialization stream begins to execute. First, the toolbar window is created as a hidden window. This is followed by the creation of the MDI frame window inside the client area of the application window. The menu is modified to remove no relevant selections. The toolbar background window is then created and the toolbar is associated with the background.

When the system is eventually initialized, it is left in a state where the toolbar can be immediately displayed in the reserved area upon loading an old or editing a new study. When the new OPERWINDOW is created, the Windows notifies the parent window of the activity which triggers a review of the menu selections and brings the system to the normal operating state.

Operating Flow

The operating flow of the Landfill model in the Windows environment is necessarily much less structured than that of its DOS counterpart. Because the application is not guaranteed to be the only application running or to have exclusive use of any of the computer resources, the user is not restricted to remain in the bounds of the application's control. Provisions must be made for any out of order method the user may take to manipulate the model.

This asynchronous nature also carries over to the objects the application will use to receive user input. Each element of the model's user interface is a window and, although there is a parent/child relationship with respect to the objects, they exhibit characteristics of encapsulation that causes difficulties when trying to uniformly accomplish tasks. Rather than hand crafting pieces of the entire mosaic, the programmer is left to build a complex creation from standardized parts.

Figure 2 shows a diagram of the overall functionality of the Windows version.

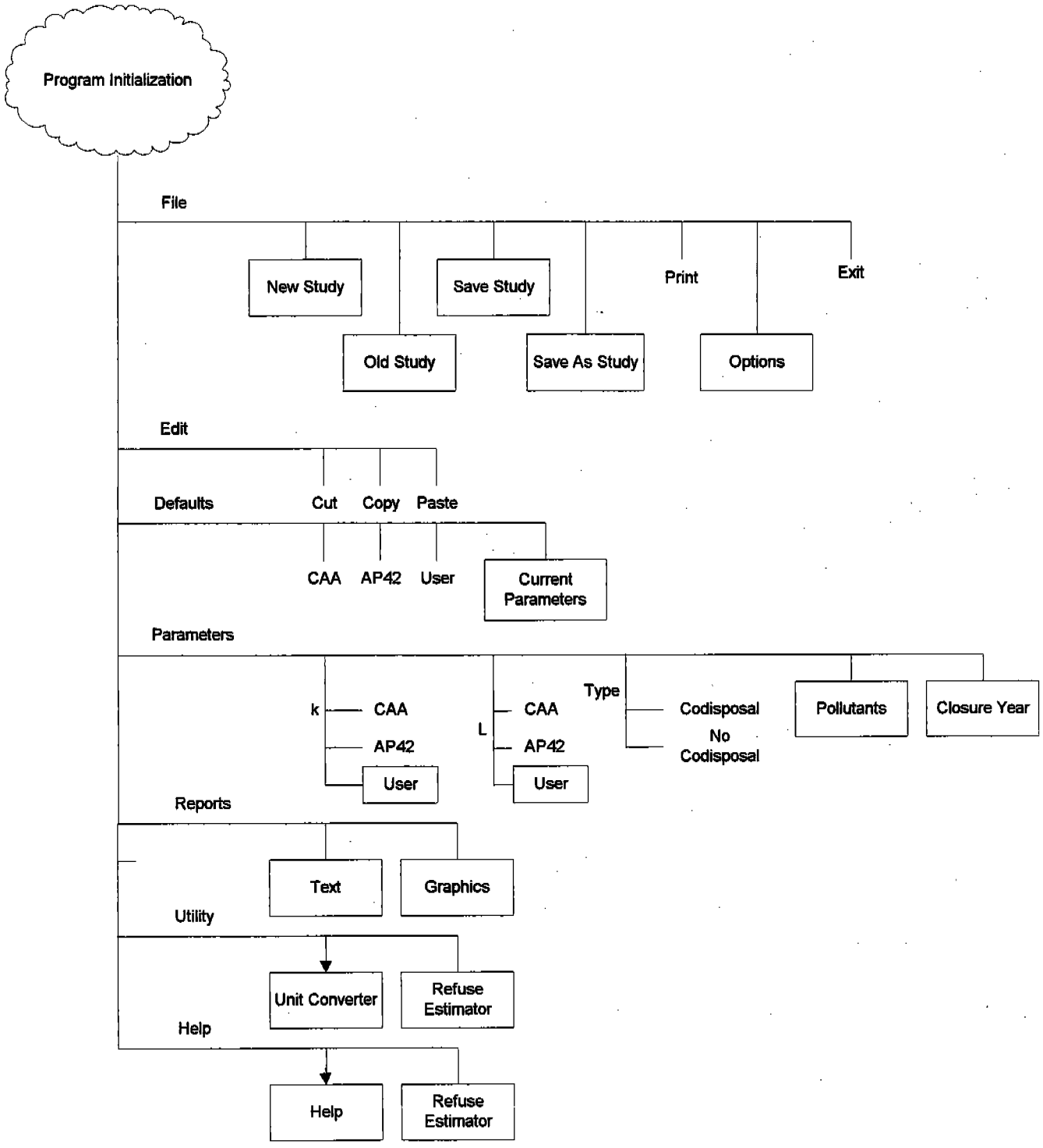


Figure 2 - Functional Diagram of Windows Model

Interoperability

As described with respect to the internal framework, the system makes heavy use of the Windows messaging system to orchestrate the behavior of the model. This communication is chiefly used in the following contexts:

- Communication between the toolbar and the operational periods window
- Communication between the MDI client window and internal client area windows
- Communication between the operational periods window and the text/graphics display windows
- Communication between the application window and the MDI children

In order to understand the behavior of the model, it is extremely important that the programmer become familiar with the scope of this communication. Unlike linearly executed programs which fail due to programming logic flaws as a result of unanticipated conditions, Windows programs have an additional possibility in that the frequently fail due to unpredictable delays or unanticipated orders of expected message delivery.

Communication between the toolbar and the operational periods windows

By far, the most difficult behavior to track is the interrelationship of the toolbar and the operational periods window. This is largely due to the fact that a single instance of the toolbar has to operate with a multiple number of operational periods windows containing different data. Orchestrating this process is a cooperative process between the operating periods window and the toolbar.

When an operational periods window appears, is brought into focus by the user, or looses focus to another window, the MDI client window is sent a message indicating whether it has gained whether it has gained or lost the input focus. The operating periods window uses this message as an opportunity to coordinate with the toolbar and insure the toolbar data and the internal data structures are synchronized. This process is largely handled by the window gaining focus and occurs in the following manner:

- The operating periods window sends a request to the application window asking for the handle of the active toolbar. The application window coordinates the creation and display of this toolbar for all MDI clients.
- The operating periods window sends a message to the toolbar to synchronize the memory structures of the window loosing focus with those currently displayed.
- The operating periods window sends a message to the toolbar to initialize its displayed data with the memory structures of the window gaining focus.

When a new operating periods window gains MDI focus, keyboard focus is actually redirected toward the active tool. The toolbar processes keyboard messages for most keys; however, when appropriate, the toolbar redirects position keys (up, down, page up, page down) to the operating periods window for processing. In the circumstance where the operating periods handles a keystroke, the window reciprocates with a message indicating which period is selected so the toolbar may reflect the correct data (as well as validate the current data).

The toolbar allows editing of data that governs that displayed in the operating periods window. The user may revise the period of operation as well as an individual period. The toolbar takes the responsibility of informing the operating period window of any change via customized messages for each purpose.

Communication between the MDI client window and internal client area windows

Most of the MDI client windows in the landfill model are composed of the MDI client itself with internal windows. The MDI client itself takes much of the responsibility of serving as the central message handler for the entire framework and passes messages to internal windows as necessary. Many of the messages are actually handled by internal windows.

Communication between the operational periods window and the text/graphics display windows

Because the model is structured such that the user may interactively change data that results in a change in the emissions, a communication method is necessary to insure that these data changes are reflected in all display clients.

For a given landfill study, there is only one MDI operating periods client which takes the primary role of orchestrating changes. If any of the data is revised, it is reflected in this window and the window takes the responsibility of informing all other clients based on this data to recalculate and redisplay the new results. The unifying thread that makes this possible is the model parameters data that is tagged to all MDI clients. The operating periods window informs all windows which have this data tag by sending an AM_VALUESCHANGED message. This process is initiated by the active toolbar, which accepts the change from the user.

Communication between the application window and the MDI children

The application window serves the role of coordinating the individual elements within the model in response to user requests. As a result, it performs the major initialization tasks which make the individual elements possible. Many of these roles require direct communication with the individual model elements using the messaging system.

Several of the menu items that are coordinated by the application window actually result in direct messages to MDI clients. This is mostly due to the fact that these requests only have validity within the context of the currently selected client. Such items include:

- Printing/saving a study or results display (graphics or text); and,
- Cutting/Pasting data within the operating periods window.

Because of its role as the initiator of the processes, it serves as the point of contact for major components. In particular, it handles the display and serves as the major supplier of the handle for the toolbar. Also, when a study is loaded or the results are displayed, it orchestrates the process of creating MDI clients and insures that data is populated *at the right time*. All of these process requires direct messages between the application window and the individual elements themselves.

Module Descriptions

Appendix E contains code listings for all modules used in the system. This section provides a succinct description of each module and contents. The source code listings provided contain documentation which expands on these discussions.

Landwin.c
This module contains the main entry point for the system, initialization routines, the main message loop, and the menu handler.

Loaddef.c
This module handles the loading/saving of default information from the resource table and the initialization files

Study.c

This module contains the code necessary to load and save a study file. This also serves as the coordination point for the creation of the operating parameters window and sets up the memory structures associated with it.

Options.c
This module handles the entry of model options (default mode and the years past the closure date for which the model will calculate emissions).

Opertool.c
This module contains the code necessary to create and manage the operating parameters toolbar.

Oper.c
This module contains the code necessary to display and manage the operating periods window.

Ap42warn.c
This module contains the dialog box code necessary to display a warning on the preliminary nature of the AP42 values when the selects them as default.

Kedit.c
This module contains the code necessary to receive user entries of the k value.

Calck.c
This module contains the user interface code necessary to manage the k calculation process.

Kcalcfn.c
This module contains the algorithm for calculating a k value.

Ledit.c
This module contains the code necessary to receive user entries of the l_0 value.

Airtoxic.c
This module contains the code necessary to allow the user to modify the emission constinuents.

Indivat.c
This module contains the code necessary to allow the user to modify the individual air toxic compound list.

Closure.c

This module contains the user interface code necessary to allow the user to input a customized closure year.

Current.c
This module displays the currently entered/selected model parameters.

Validate.c
This module performs validity checks on entered data to insure calculation can occur prior to beginning the calculation process.

Calcmod.c
This module contains the calculation algorithms to estimate emissions.

Makegph.c

This module initiates the process of producing a graphical report.

Graphics.c
This module creates the graphical report.

Doclip.c
This module allows for creation of the graphic to the Windows clipboard.

Doprint.c
This module allows printing of the graphic output.

Dowmf.c
This module allows saving a graphic as a Windows Metafile.

Makerpt.c
This module initiates the process of producing the text report.

Bldrpt.c
This module produces the physical report.

Textrpt.c
This module allows for the display of the text report as an MDI client.

Textfile.c
This module produces the text report output to file.

Textprnt.c
This module handles printing of the text report.

Rprtsel.c
This module contains the code necessary to allow the user to select a reported emission type.

About.c
This module handles the display of the about box.

Estimate.c
This module handles the display of the refuse estimate dialog.

Landcust.c
This module contains the code necessary to create that background custom control for the toolbar and the operating periods window.

Subclass.c
This module contains the subclass management routines and the bevel code for the edit boxes.

Utility.c
This module contains the utility routines.

Str_ops.c
This module contains miscellaneous string handling functions.

Include Files

Landwin.h
This include file contains function prototypes.

Datadef.h
This include file contains the basic data structure definitions.

Landoper.h
This include file contains the definition of the operational data structure.

Graphics.h
This include file contains the definitions of the graphics data parameters and function prototypes.

Results.h
This include file contains the definition of the results data structure.

Textrpt.h
This include file contains the definition of the report data structures and defines for managing the report process.

Helpdef.h
This include file contains the definition of the context sensitive help identifiers.

Landres.h
This include file contains the resource identifiers and the custom Windows message types.

Str_ops.h
This include file contains the prototypes for the string handling functions.

Appendix A

DOS Version NMake File

```
#               Make file for the LANDFILL PROGRAM - 04/04/90
#
#                       Written by RADIAN CORPORATION
#                        Research Triangle Park, NC
#
# Modified by RLB (Radian) 12/93 - Included support for a portable
#           working environment during the development and allowing
#           for different origins of the libraries and includes.

# Set the macro definitions for the executable and the link
# libraries here.

exec = landfill.exe
map = xxx.map
debug = /Z7 /DNOMEMCHK

# Different directories for the compilation headers and libraries
# depending on the system.

!IF "$(WORKSTATION)" == "PORTABLE"

BIN=d:\msvc\bin
INCLUDE=d:\msvc\include
LIB=d:\msvc\lib

cscape_include = d:\proglang\old_cscp\include\
cscape_lib = d:\proglang\old_cscp\lib\

!ELSE

BIN=F:\msvc\bin
INCLUDE = F:\msvc\include
LIB=F:\msvc\lib

cscape_include = c:\proglang\old_cscp\include\
cscape_lib = c:\proglang\old_cscp\lib\

!ENDIF

# Compilation command with options - /Zi /Zd are debug options
cc = $(BIN)\cl /c /AL /I$(cscape_include) /I$(INCLUDE)


# Link options - /CO is the debug option
link_opts = /STACK:16000 /SE:256

link = $(BIN)\link $(link_opts)


$(exec) : main.obj editmenu.obj mainmenu.obj new-old.obj credits.obj \
          speckey.obj fnsusdbl.obj misc_utl.obj landfill.h landuser.h \
          methrate.obj methcap.obj chemcomp.obj editmain.obj airtoxmn.obj \
          periods.obj fnsusint.obj kcalcmn.obj kcalcmen.obj kedit.obj \
          disprslt.obj tblemenu.obj studname.obj closure.obj vferror.obj \
          fndosstr.obj warnings.obj calcmain.obj scr_view.obj help_scr.obj \
          summary.obj selector.obj config.obj exit.obj savemenu.obj  \
          savestud.obj fnticker.obj strmview.obj fndosfil.obj getout.obj \
          do_graph.obj keyhelp.obj buildbox.obj flselect.obj findfls.obj \
          fnselect.obj
          @SET LIB=$(LIB)
          @echo main editmenu mainmenu new-old credits + > landfill.lnk
          @echo speckey fnsusdbl misc_utl methrate methcap + >> landfill.lnk
          @echo chemcomp editmain airtoxmn periods fnsusint + >> landfill.lnk
          @echo kcalcmn kcalcmen kedit disprslt tblemenu studname + >> landfill.
lnk
          @echo closure vferror fndosstr warnings calcmain scr_view + >> landfil
```



```
l.lnk
          @echo help_scr summary selector config exit savemenu + >> landfill.lnk
          @echo savestud fnticker strmview fndosfil getout do_graph + >> landfil
l.lnk
          @echo keyhelp buildbox flselect findfls fnselect >> landfill.lnk
          @echo $(exec),$(map) >> landfill.lnk
          @echo $(cscape_lib)m5lcscap.lib pgchart.lib $(link_opts) ; >> landfil
.lnk

          @$(link) @landfill.lnk

          @del landfill.lnk

main.obj : main.c landfill.h defaults.h

do_graph.obj : do_graph.c landfill.h

keyhelp.obj : keyhelp.c keyhelp.h

config.obj : config.c landfill.h helpasgn.h

editmenu.obj : editmenu.c landfill.h

editmain.obj : editmain.c landfill.h

calcmain.obj : calcmain.c landfill.h defaults.h

summary.obj : summary.c landfill.h landuser.h defaults.h

scr_view.obj : scr_view.c landfill.h defaults.h

strmview.obj : strmview.c landfill.h defaults.h

selector.obj : selector.c landfill.h

mainmenu.obj : mainmenu.c

exit.obj : exit.c

savemenu.obj : savemenu.c

savestud.obj : savestud.c

buildbox.obj : buildbox.c

help_scr.obj : help_scr.c

studname.obj : studname.c

disprslt.obj : disprslt.c

tblemenu.obj : tblemenu.c

speckey.obj : speckey.c

fnsusdbl.obj : fnsusdbl.c

fnticker.obj : fnticker.c

fnsusint.obj : fnsusint.c

fndosstr.obj : fndosstr.c

fndosfil.obj : fndosfil.c

fnselect.obj : fnselect.c
```

```
getout.obj : getout.c landuser.h

misc_utl.obj : misc_utl.c

credits.obj : credits.c landfill.h

findfls.obj : findfls.c landfill.h

flselect.obj : flselect.c landfill.h

new-old.obj : new-old.c

methrate.obj : methrate.c defaults.h

methcap.obj : methcap.c defaults.h

kedit.obj : kedit.c defaults.h

chemcomp.obj : chemcomp.c landfill.h defaults.h

airtoxmn.obj : airtoxmn.c landfill.h landuser.h defaults.h

periods.obj : periods.c landfill.h landuser.h defaults.h

closure.obj : closure.c landfill.h landuser.h defaults.h

kcalcmen.obj : kcalcmen.c landfill.h landuser.h defaults.h

kcalcmn.obj : kcalcmn.c landfill.h constant.h defaults.h

vferror.obj : vferror.c vfmsgs.h

warnings.obj : warnings.c
```

Appendix B

DOS Version Code Listings

```
/* ========================================================================= */
/*                                COLORDFS.H                                 */
/* ========================================================================= */
/*  This header file contains the global color information for the menus     */
/* ========================================================================= */
/*                           Created under Contract                          */
/*                                    to                                     */
/*                             EPA/AEERL/GECD/EMB                            */
/*                                                                           */
/*                                    by                                     */
/*                                                                           */
/*                             Radian Corporation                            */
/*                           Research Triangle Park, NC                      */
/* ========================================================================= */
/* ========================================================================= */
/*                             Revision History                              */
/* ========================================================================= */
/*    06/17/90    rlb        Initial Implementation                          */
/* ========================================================================= */

unsigned char menu_color = (BLUE << 4) | CYAN;
unsigned char warning_color = (CYAN << 4) | RED;
unsigned char error_color = (RED << 4) | WHITE;
unsigned char info_color = (CYAN << 4) | BLACK;
unsigned char keyhelp_color = (BLUE << 4) | CYAN;
unsigned char field_color = (WHITE << 4) | BLACK;
```

```
/* ========================================================================== */
/*                                  COLORS.H                                  */
/* ========================================================================== */
/*  This include file assigns the appropriate bit sequences to colors         */
/* ========================================================================== */
/*                             Created under Contract                         */
/*                                       to                                   */
/*                               EPA/AEERL/GECD/EMB                           */
/*                                                                            */
/*                                       by                                   */
/*                                                                            */
/*                                Radian Corporation                          */
/*                             Research Triangle Park, NC                     */
/* ========================================================================== */
/* ========================================================================== */
/*                                Revision History                            */
/* ========================================================================== */
/*    04/17/90   rlb        Initial Implementation                            */
/* ========================================================================== */

#define COLORS_DEFINED

#define BLACK     0
#define BLUE      1
#define GREEN     2
#define CYAN      3
#define RED       4
#define MAGENTA   5
#define BROWN     6
#define WHITE     7

#define INTENSE   8
#define BLINK     128

extern unsigned char menu_color ;
extern unsigned char warning_color ;
extern unsigned char error_color ;
extern unsigned char info_color ;
extern unsigned char keyhelp_color ;
extern unsigned char field_color;
```

```
/* ========================================================================== */
/*                                CONSTANT.H                                  */
/* ========================================================================== */
/*    This include file defines math constants for use in calculation.        */
/* ========================================================================== */
/*                            Created under Contract                          */
/*                                    to                                      */
/*                              EPA/AEERL/GECD/EMB                            */
/*                                                                            */
/*                                    by                                      */
/*                                                                            */
/*                              Radian Corporation                            */
/*                           Research Triangle Park, NC                       */
/* ========================================================================== */
/* ========================================================================== */
/*                              Revision History                              */
/* ========================================================================== */
/*    04/17/90   rlb       Initial Implementation                             */
/* ========================================================================== */


#ifndef CONSTANTS
#define CONSTANTS

#define PI  3.14159265358979323846

/* ========================================================================== */
/*              R - Universal Gas Constant  (J / (mole K))                    */
/* ========================================================================== */

#define R   8.3144

/* ========================================================================== */
/*    Multiplication Factor to convert Cubic Ft to Cubic Meters               */
/* ========================================================================== */

#define FT3_TO_M3      0.028316846

#endif
```

```
/* ========================================================================== */
/*                                DEFAULTS.H                                  */
/* ========================================================================== */
/*    This include file defines all of the default values for the             */
/*    parameters used in the program.                                         */
/* ========================================================================== */
/*                           Created under Contract                           */
/*                                    to                                      */
/*                             EPA/AEERL/GECD/EMB                             */
/*                                                                            */
/*                                    by                                      */
/*                                                                            */
/*                             Radian Corporation                             */
/*                         Research Triangle Park, NC                         */
/* ========================================================================== */
/* ========================================================================== */
/*                             Revision History                               */
/* ========================================================================== */
/*    04/17/90   rlb       Initial Implementation                             */
/* ========================================================================== */


#ifndef DEFAULTS_DEFINED
#define DEFAULTS_DEFINED

#define DEFAULT_YEARS_PRIOR      10

#define DEFAULT_CH4              50.0
#define DEFAULT_CO2              50.0

#define DEFAULT_REGULATORY_L0    6000.0
#define DEFAULT_REGULATORY_K     0.05
#define DEFAULT_REGULATORY_NMOC  4000.0

#define DEFAULT_AP42_K           0.04
#define DEFAULT_AP42_L0          4411
#define DEFAULT_AP42_NMOC_CODIS  2420
#define DEFAULT_AP42_NMOC_NOCO   595

#define DEFAULT_WELL_DEPTH       0.0
#define DEFAULT_INFLUENCE_RAD    0.0
#define DEFAULT_REFUSE_DENSITY   0.018
#define DEFAULT_DECOMP_FRACTION  1.0
#define DEFAULT_FLOW_RATE        0.0
#define DEFAULT_REFUSE_AGE       0.0

#define DEFAULT_TEMP             293.0
#define DEFAULT_PRESSURE         1.01325E5

#define DEFAULT_NMOC_MOLE_WT     86.18    /*  Molecular Wt of NMOC is  */
                                          /*  established assuming it  */
                                          /*  is HEXANE.               */

#define YEARS_PASSED_CLOSURE     50
#define MAX_YEARS_PAST_CLOSURE 200
#define MAX_DURATION             300
#define MAX_AIR_TOXICS           60

#define LINES_PER_PAGE           66
#define PAGE_WIDTH               80

#endif
```

```
/* ========================================================================= */
/*                                HELPASGN.H                                 */
/* ========================================================================= */
/*    This include file assigns help numbers to all of the help tags.        */
/*      Help tags are all caps versions of the menu function names.          */
/* ========================================================================= */
/*                           Created under Contract                          */
/*                                     to                                    */
/*                             EPA/AEERL/GECD/EMB                            */
/*                                                                           */
/*                                     by                                    */
/*                                                                           */
/*                             Radian Corporation                            */
/*                          Research Triangle Park, NC                       */
/* ========================================================================= */
/* ========================================================================= */
/*                             Revision History                              */
/* ========================================================================= */
/*    04/17/90   rlb       Initial Implementation                            */
/* ========================================================================= */

#ifndef HELP_DEFINED
#define HELP_DEFINED

#define MAIN_MENU              1
#define STUDY_TYPE             2
#define GET_STUDY_NAME         3
#define EDIT_MENU              4
#define EDIT_CHEM_COMP         5
#define AIR_TOXICS_MENU        6
#define K_SELECTION_MENU       7
#define EDIT_K_VALUE           8
#define CALCULATE_K_MENU       9
#define L0_SELECTION_MENU      10
#define EDIT_PERIODS           11
#define DISPLAY_RESULTS_MENU   12
#define TABULAR_REPORT_MENU    13
#define ON_SCREEN_GRAPHIC      14
#define REPORT_SELECTOR        15
#define SHOW_RESULTS           16
#define CONFIGURE_SYSTEM       17
#define DISPLAY_SAVE_MENU      18
#define GET_OUTPUT_FILENAME    19
#define DISPLAY_CLOSURE_YEAR   20
#define FILE_SELECTOR          21

extern char *mainmenu_help_string, *chemcomp_help_string,
            *airtoxmn_help_string, *generic_help_string,
            *generic_menu_help_string, *studname_help_string,
            *config_help_string, *continue_help_string,
            *tabular_report_key_help, *any_key_string,
            *abortable_menu_help_string, *abortable_warning_help_string,
            *no_default_help_string;

#endif
```

```
/* ========================================================================= */
/*                                KEYHELP.H                                  */
/* ========================================================================= */
/*          This include file defines the text for the key help.             */
/* ========================================================================= */
/*                            Created under Contract                         */
/*                                      to                                   */
/*                              EPA/AEERL/GECD/EMB                           */
/*                                                                           */
/*                                      by                                   */
/*                                                                           */
/*                              Radian Corporation                           */
/*                           Research Triangle Park, NC                      */
/* ========================================================================= */
/* ========================================================================= */
/*                              Revision History                             */
/* ========================================================================= */
/*    04/17/90   rlb       Initial Implementation                            */
/* ========================================================================= */

char *continue_help_string = "F10 = Continue";

char *any_key_string = "Any Key = Continue";

char *mainmenu_help_string =
"F1 = Help              ESC = Exit             Enter = Select Item";

char *studname_help_string =
"F1 = Help       F2 = Select File       ESC = Abort       Enter = Accept Name";

char *generic_menu_help_string =
"F1 = Help           ESC = Previous           Enter = Select Item";

char *abortable_menu_help_string =
"F1 = Help             ESC = Abort            Enter = Select Item";

char *abortable_warning_help_string =
"ESC = Abort           Any Key = Continue";

char *airtoxmn_help_string =
"F1=Help    F2=Reset Default    F5/F6=Insert/Delete    ESC=Abort    F10=Accept";

char *chemcomp_help_string =
"F1=Help   SPACE=Toggle Air Toxics   F2=Reset Default   ESC=Abort   F10=Accept";

char *config_help_string =
"F1 = Help     SPACE = Toggle Printer Choices     ESC = Abort     F10 = Accept";

char *generic_help_string =
"F1 = Help       F2 = Reset Default       ESC = Abort       F10 = Accept";

char *no_default_help_string =
"F1 = Help                   ESC = Abort                   F10 = Accept";

char *tabular_report_key_help =
"F1 = Help          / Pg = Move through Text         ESC = Quit";
```

```
/* ========================================================================== */
/*                                LANDFILL.H                                  */
/* ========================================================================== */
/*    This include file defines the major data structures used in the         */
/*    program.                                                                */
/* ========================================================================== */
/*                           Created under Contract                           */
/*                                    to                                      */
/*                             EPA/AEERL/GECD/EMB                             */
/*                                                                            */
/*                                    by                                      */
/*                                                                            */
/*                             Radian Corporation                             */
/*                          Research Triangle Park, NC                        */
/* ========================================================================== */
/* ========================================================================== */
/*                           Revision History                                 */
/* ========================================================================== */
/*    04/17/90    rlb       Initial Implementation                           */
/*    01/21/94    rlb       Added NO_AT to defines.  Also defining           */
/*                             macros for defaults mode.                      */
/* ========================================================================== */

#define TITLE_STRING "Version 2.0"

#ifndef REGULATORY_MODE
#define REGULATORY_MODE 0
#define AP_MODE         1
#endif

#ifndef LANDFILL_PARMS
#define LANDFILL_PARMS

struct landfill_parms {
                      int year_opened, current_year, year_closed;
                      double capacity, avg_acceptance_rate, last_rate;
                      int number_of_periods;
                      double *refuse_in_place;
                      };

#endif



#ifndef AIRTOX_STRUCTURE
#define AIRTOX_STRUCTURE

#define NOCODISPOSAL 0
#define CODISPOSAL   1
#define NO_AT        2

struct air_toxic_structure {
                           char chemical_name[50];
                           double molecular_wt;
                           double concentration[2];
                           };

#endif


#ifndef MODEL_PARMS
#define MODEL_PARMS

struct model_parms {
                   double l0, k, nmoc, ch4, co2;
                   int include_air_toxics;
```



```
                   struct air_toxic_structure *air_toxics;
                   int number_air_toxics;
                   };

#endif


#ifndef K_CALC_PARMS
#define K_CALC_PARMS

struct k_calc_parms {
                    double well_depth, radius, density;
                    double fraction, flow_rate, refuse_age;
                    };

#endif


#ifndef MODEL_RESULTS_DEFINED
#define MODEL_RESULTS_DEFINED

struct model_results {
                     int year_count;
                     int air_toxics_count;
                     int *year_list;
                     double *refuse_data;
                     double *annual_ch4;
                     double *annual_co2;
                     double *annual_nmoc;
                     double **annual_air_toxics;
                     };

#endif


#ifndef SYSTEM_PARMS
#define SYSTEM_PARMS

struct system_parms {
                    char working_dir[256];
                    char landfill_dir[256];
                    int print_device;
                    int default_mode;
                    int years_passed_closure;
                    };

extern struct system_parms config;

#endif


#ifndef MISC_ROUTINES_DEFINED
#define MISC_ROUTINES_DEFINED

extern int pad_string(char *, int);
extern char *midstr(char *, int, int);
extern int capital_string(char *);
extern void show_mem_error(char *, char *);

extern void verify_error(int);

#endif
```

```
/* ========================================================================== */
/*                                LANDFILL.H                                  */
/* ========================================================================== */
/*    This include file defines the user interface information for the        */
/*    cscape library                                                          */
/* ========================================================================== */
/*                           Created under Contract                           */
/*                                    to                                      */
/*                             EPA/AEERL/GECD/EMB                             */
/*                                                                            */
/*                                    by                                      */
/*                                                                            */
/*                             Radian Corporation                             */
/*                          Research Triangle Park, NC                        */
/* ========================================================================== */
/* ========================================================================== */
/*                             Revision History                               */
/* ========================================================================== */
/*    04/17/90   rlb       Initial Implementation                             */
/* ========================================================================== */

#ifndef SPECIAL_FIELDS
#define SPECIAL_FIELDS

extern  void    susdbl_fenter(sed_type);
extern  boolean susdbl_fexit(sed_type);
extern  void    susdbl_fkey(sed_type);
extern  void    susdbl_senter(sed_type, int);
extern  void    susdbl_sexit(sed_type, int);

extern  struct  field_funcs susdbl_funcs;


extern  void    susint_fenter(sed_type);
extern  boolean susint_fexit(sed_type);
extern  void    susint_fkey(sed_type);
extern  void    susint_senter(sed_type, int);
extern  void    susint_sexit(sed_type, int);

extern  struct  field_funcs susint_funcs;


extern  void    dosstr_fenter(sed_type);
extern  boolean dosstr_fexit(sed_type);
extern  void    dosstr_fkey(sed_type);
extern  void    dosstr_senter(sed_type, int);
extern  void    dosstr_sexit(sed_type, int);

extern  struct  field_funcs dosstr_funcs;


extern  void    dosfile_fenter(sed_type);
extern  boolean dosfile_fexit(sed_type);
extern  void    dosfile_fkey(sed_type);
extern  void    dosfile_senter(sed_type, int);
extern  void    dosfile_sexit(sed_type, int);

extern  struct  field_funcs dosfile_funcs;


extern  void      myselect_fenter(sed_type);
extern  boolean myselect_fexit(sed_type);
extern  void      myselect_fkey(sed_type);
extern  void      myselect_senter(sed_type, int);

extern  struct  field_funcs myselect_funcs;
```



```
#endif
```

```
/* ========================================================================== */
/*                                LANDFILL.H                                  */
/* ========================================================================== */
/*    This include file defines the individual error messages displayed       */
/*    by the verification error screen.                                       */
/* ========================================================================== */
/*                           Created under Contract                           */
/*                                    to                                      */
/*                             EPA/AEERL/GECD/EMB                             */
/*                                                                            */
/*                                    by                                      */
/*                                                                            */
/*                             Radian Corporation                             */
/*                          Research Triangle Park, NC                        */
/* ========================================================================== */
/* ========================================================================== */
/*                             Revision History                               */
/* ========================================================================== */
/*    04/17/90   rlb       Initial Implementation                             */
/* ========================================================================== */

struct messages
               {
               char string1[40], string2[40], string3[40];
               };


struct messages error_msg[] = {
   {"Refuse in place is less", "than data in previous", "years"},
   {"Final value for refuse", "in place must be", "greater than zero"},
   {"Largest value cannot", "exceed permitted capacity", ""},
   {"Current year must be", "greater than the year", "opened"},
   {"Capacity cannot be zero", "", ""},
   {"Year of closure must be", "at least one year beyond", "the current year"},
   {"Operation cannot exceed", "internal limit of", "300 years"},
   {"Methane percentage", "cannot exceed", "100 percent"},
   {"Number of air toxics", "cannot exceed an internal", "limit of 20"},
   {"Methane percentage of", "landfill gas cannot","be zero"},
   {"Chemical name must be", "specified",""},
   {"Molecular weight must", "be greater than","zero"},
   {"Calculation must not", "exceed 200 years passed", "closure."},
   {"NMOC value cannot", "exceed 1,000,000 ppmv.", ""}
                              };
```

```
/* ========================================================================== */
/*                                 AIRTOXMN.C                                 */
/* ========================================================================== */
/*  This file contains the code for displaying and altering the list of       */
/*  individual air toxics tracked by the program.                             */
/* ========================================================================== */
/*                            Created under Contract                          */
/*                                    to                                      */
/*                              EPA/AEERL/GECD/EMB                            */
/*                                                                            */
/*                                    by                                      */
/*                                                                            */
/*                              Radian Corporation                            */
/*                           Research Triangle Park, NC                       */
/* ========================================================================== */
/* ========================================================================== */
/*                              Revision History                              */
/* ========================================================================== */
/*    07/03/90   rlb       Initial Implementation                             */
/*    01/17/94   rlb       Altered to allow entry of a codisposal and         */
/*                            no codisposal emission factor.  Also,           */
/*                            specifies the printing format for double        */
/*                            values using the modification to fnsusdbl       */
/* ========================================================================== */

#include <stdio.h>
#include <cscape.h>
#include <scancode.h>
#include <string.h>
#include <ctype.h>
#include <malloc.h>
#include "landfill.h"
#include "landuser.h"
#include "helpasgn.h"
#include "defaults.h"
#include "colors.h"

/* ========================================================================== */
/*   Define window markers by the IBM ascii code representations              */
/* ========================================================================== */

#define UP_ARROW      24
#define DOWN_ARROW    25
#define SPACE         32

/* ========================================================================== */
/*                     This is the main menuing function.                     */
/*  Air_toxic_ptr is passed in as an pointer to an array pointer because      */
/*  this routine has a requirement to change the location of that list        */
/*  periodically as the list length changes and memory configurations         */
/*  change.                                                                   */
/* ========================================================================== */

int air_toxics_menu(air_toxic_ptr, number_air_toxics)
struct air_toxic_structure **air_toxic_ptr;
int *number_air_toxics;
{
extern struct system_parms config;
struct air_toxic_structure *air_toxic, *scratch_air_toxic;
menu_type   menu, choice_menu, key_menu;
sed_type    sed, choice_sed, key_sed;
int         ret, i, old_offset = 0, curr_offset,
            current_end, current_field, scratch, scratch_number;
char up_indicator = SPACE, down_indicator = SPACE;
char chemical[31];
char *ptr;
```



```
double mole_wt = 0.0;

int create_air_toxic_menu(menu_type *, sed_type *,
                          struct air_toxic_structure *);
int delete_air_toxics_menu(int, menu_type, sed_type);
int copy_air_toxic_list(struct air_toxic_structure **,
                        struct air_toxic_structure **, int);

extern void initialize_air_toxics(char *, char *,
                    struct air_toxic_structure **, int *);

   reset_insert_status();

/* ========================================================================== */
/*  Create a copy of the original air toxic list so we can change the         */
/*  information without destroying the original                               */
/* ========================================================================== */

   copy_air_toxic_list(&scratch_air_toxic, air_toxic_ptr,
                       *number_air_toxics);

   scratch_number = *number_air_toxics;


/* ========================================================================== */
/*                         Start defining the Cscape menu                     */
/* ========================================================================== */

   menu = menu_Open();

   build_scrollbox(menu, menu_color, 18, 78);

        menu_Printf(menu, "@p[1,24]Edit Air Pollutant Concentration");
        menu_Printf(menu, "@p[2,0]|@[76,-]|");
        menu_Printf(menu, "@p[3,0]|     ");
        menu_Printf(menu, "@p[3,35]Molecular");
        menu_Printf(menu, "@p[3,51]Concentration (ppm V)");
        menu_Printf(menu, "@p[4,0]|     ");
        menu_Printf(menu, "@p[4,14]Chemical");
        menu_Printf(menu, "@p[4,35] Weight    ");
        menu_Printf(menu, "@p[4,49]No Codisposal  Codisposal");
        menu_Printf(menu, "@p[5,0]|--@[74,-]|");

   menu_Printf(menu, "@p[16,1]@fp[#]", &up_indicator, &char_funcs);
   menu_Printf(menu, "@p[6,1]@fp[#]",  &down_indicator, &char_funcs);

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, menu_color, menu_color, menu_color);
   sed_SetPosition(sed, 3, 1);

   sed_Push(sed);

/* ========================================================================== */
/*  Before you draw the menu on the screen, check to see if the menu list     */
/*  is longer than the available window.  If so, set the down indicator       */
/*  to its on state and redraw the screen.                                    */
/* ========================================================================== */

   if (*number_air_toxics > 12) down_indicator = DOWN_ARROW;

   sed_Repaint(sed);
   show_key_help(&key_menu, &key_sed, airtoxmn_help_string);

   sed_SetActive(sed, TRUE);
```

```
/* ========================================================================= */
/*  Check to see if there is any air toxics in the list.  If so, create a    */
/*  menu with them, draw it on the screen, and set it for entry.  This is    */
/*  necessary because Cscape does not like menus with 0 entries.             */
/* ========================================================================= */

   if (scratch_number)
      {
      create_air_toxics_menu(&choice_menu, &choice_sed, scratch_air_toxic,
                             scratch_number);

      sed_Repaint(choice_sed);
      sed_SetActive(choice_sed, TRUE);
      sed_GotoFirstField(choice_sed);

      sed_DoFieldSenter(choice_sed, sed_GetFieldNo(choice_sed));
      sed_DoFieldFenter(choice_sed, sed_GetFieldNo(choice_sed));
      }

/* ========================================================================= */
/*                          This is the Main Loop                            */
/*     It will continue until the user presses FN10 or the ESC key.         */
/* ========================================================================= */

   do
      {

/* ========================================================================= */
/*  If there are any entries in the air toxic list, check to see if there    */
/*  are too many for the menu screen, set the more indicators, and begin     */
/*  accepting field entry information.  If there are no entries, just get    */
/*  the keys hit.                                                            */
/* ========================================================================= */

      if (scratch_number)
         {
         curr_offset = sed_GetYoffset(choice_sed);

         if ((curr_offset) && (scratch_number > 11))
            up_indicator = UP_ARROW;
         else
            up_indicator = SPACE;

         if (scratch_number - curr_offset > 11)
            down_indicator = DOWN_ARROW;
         else
            down_indicator = SPACE;

         if ((curr_offset != old_offset) || (ret == FN5) || (ret = FN6))
            {
            sed_RepaintFields(sed);
            old_offset = curr_offset;
            }

         ret = accept_menu_choices(choice_sed);
         }
      else
         ret = kb_Read();

/* ========================================================================= */
/*                          Evaluate key entered                             */
/* ========================================================================= */

      switch (ret)
         {
```



```
/* ========================================================================= */
/*  FN5 - Add a new air toxic.                                               */
/*                                                                           */
/*  First, check to see if they have maxed out on air toxics.  If not,       */
/*  call the routine to prompt for the information and check to see if       */
/*  they terminated with an F10 key (if not, stop).  If a list exists,       */
/*  delete it first.  Then create a new list and put the new info in it.     */
/*  Since we have got to have at least one entry now, recreate the menu,     */
/*  display it, and start accepting field entry from it.                     */
/* ========================================================================= */
         case FN5:
            if (scratch_number + 1 > MAX_AIR_TOXICS)
               {
               verify_error(8);
               break;
               }

            chemical[0] = '\0';
            mole_wt = 0.0;

            if (add_air_toxic(chemical, &mole_wt))
               {
               if (scratch_number)
                  delete_air_toxics_menu(scratch_number, choice_menu,
                                         choice_sed);

               scratch_air_toxic = (struct air_toxic_structure *)
                                   realloc(scratch_air_toxic,
                                     (scratch_number + 1) *
                                       sizeof(struct air_toxic_structure));

               strcpy(scratch_air_toxic[scratch_number].chemical_name, chemical)
;
               scratch_air_toxic[scratch_number].molecular_wt = mole_wt;
               scratch_air_toxic[scratch_number].concentration[NOCODISPOSAL] = 0
.0;
               scratch_air_toxic[scratch_number].concentration[CODISPOSAL] = 0.0
;

               create_air_toxics_menu(&choice_menu, &choice_sed,
                                      scratch_air_toxic, scratch_number + 1);

               sed_SetActive(choice_sed, TRUE);
               sed_GotoField(choice_sed, 4 * (scratch_number)++ + 2);

               sed_Repaint(choice_sed);

               sed_DoFieldSenter(choice_sed, sed_GetFieldNo(choice_sed));
               sed_DoFieldFenter(choice_sed, sed_GetFieldNo(choice_sed));
               }

            break;

/* ========================================================================= */
/*  FN6 - Delete an air toxic.                                               */
/*                                                                           */
/*  First, check to see if there is anything to delete. If so, get the       */
/*  current field number so we can determine which entry to delete.          */
/*  Next, delete the menu (which must exist since entries exist) and,        */
/*  if there is more than 1 entry in the list, call the function to loose    */
/*  the entry we are sitting on (since there are 3 fields for each entry     */
/*  and we are sitting on the 3rd of 3 fields, the entry must be the          */
/*  current field divided by 3), otherwise, loose the entire list.  Now,     */
/*  check to make sure there is still entries in the list now and, if so,    */
/*  build the menu, draw it, and get field input.                            */
/* ========================================================================= */
```

```
        case FN6:
           if (scratch_number)
              [
              current_field = sed_GetFieldNo(choice_sed);
              delete_air_toxics_menu(scratch_number, choice_menu,
                                     choice_sed);

              if (scratch_number > 1)
                 delete_airtox_entry(&scratch_air_toxic, current_field / 4,
                                     scratch_number);
              else
                 {
                 free(scratch_air_toxic);
                 scratch_air_toxic = NULL;
                 }

              (scratch_number)--;
              }

           if (scratch_number)
              [
              create_air_toxics_menu(&choice_menu, &choice_sed,
                                     scratch_air_toxic, scratch_number);
              sed_SetActive(choice_sed, TRUE);
              scratch = current_field - 4;
              sed_GotoField(choice_sed, (scratch <= 1) ? 2 : scratch);
              sed_Repaint(choice_sed);
              ]

           break;

/* ========================================================================== */
/*  FN2 - Reset the Air Toxic List to Defaults                                */
/*                                                                            */
/*  Pop up a warning screen.  If they tell you to go ahead, delete the        */
/*  menu (if there is one) and free the memory.  Reinitialize the list        */
/*  by re-reading the air toxics file and build the new menu.                 */
/* ========================================================================== */
        case FN2:
           if (warn_about_reset())
              {
              if (scratch_number)
                 delete_air_toxics_menu(scratch_number, choice_menu,
                                        choice_sed);

              if (scratch_air_toxic != NULL) free(scratch_air_toxic);
              initialize_air_toxics(config.landfill_dir, "airtoxic.txt",
                                    &scratch_air_toxic, &scratch_number);

              if (scratch_number)
                 [
                 create_air_toxics_menu(&choice_menu, &choice_sed,
                                        scratch_air_toxic, scratch_number);
                 sed_SetActive(choice_sed, TRUE);
                 sed_GotoFirstField(choice_sed);
                 sed_Repaint(choice_sed);
                 }
              ]

           break;

/* ========================================================================== */
/*  FN1 - Show the Help Information for this screen                           */
/* ========================================================================== */

        case FN1:
```



```
              compile_help(AIR_TOXICS_MENU);
              break;
           }
        ]
     while ((ret != FN10) && (ret) && (ret != ESC));

/* ========================================================================== */
/*                                 End of Loop.                               */
/*  Note: there are 3 conditions being tested.  This is required because      */
/*  two different routines are being called as necessary to get input         */
/*  from the menu.  sed_Go and kb_Read both return FN10.  However, sed_Go     */
/*  returns 0 when ESC is return while kb_Read returns ESC.                   */
/* ========================================================================== */

/* ========================================================================== */
/*  If the loop was exited with entries in the list, make sure the fields     */
/*  have all been converted and delete the menu.                              */
/* ========================================================================== */
     if (scratch_number)
        [
        sed_DoFieldSexit(choice_sed, sed_GetFieldNo(choice_sed));

        sed_SetActive(choice_sed, FALSE);
        delete_air_toxics_menu(scratch_number, choice_menu, choice_sed);
        ]

/* ========================================================================== */
/*  The loop was exited because FN10 was pressed.  That means the data        */
/*  was accepted and we should copy the scratch data areas we created         */
/*  into the real ones.                                                       */
/* ========================================================================== */

     if (ret == FN10)
        {
        *number_air_toxics = scratch_number;

/* ========================================================================== */
/*  Free up the old list's memory and create a new list with the new          */
/*  number of entries.                                                        */
/* ========================================================================== */

        if (*air_toxic_ptr != NULL)
           {
           free(*air_toxic_ptr);
           *air_toxic_ptr = NULL;
           }

        copy_air_toxic_list(air_toxic_ptr, &scratch_air_toxic,
                            *number_air_toxics);
        }

/* ========================================================================== */
/*                    Deallocate the scratch lists memory.                    */
/* ========================================================================== */

     if (scratch_number) free(scratch_air_toxic);

/* ========================================================================== */
/*          Delete the menu and return with the key they exited on.           */
/* ========================================================================== */
     sed_SetActive(sed, FALSE);

     destroy_key_help(key_menu, key_sed);
     sed_Pop(sed);

     menu_Close(menu);
```

```
   sed_Close(sed);

   return(ret);
}



/* ========================================================================== */
/*  This function makes space for and copies an original air toxics list  */
/*  to a new list.                                                          */
/* ========================================================================== */

int copy_air_toxic_list(dest_ptr, source_ptr, count)
struct air_toxic_structure **dest_ptr, **source_ptr;
int count;
{
struct air_toxic_structure *dest_list, *source_list;
int i;

   source_list = *source_ptr;

   if (count == 0)
      {
      *dest_ptr = NULL;
      return(0);
      }

   dest_list = *dest_ptr = (struct air_toxic_structure *)
               calloc(count, sizeof(struct air_toxic_structure));

   if (dest_list == NULL)
      show_mem_error("copy_air_toxic_list", "dest_ptr");

   for (i = 0 ; i < count ; i++)
      {
      strcpy(dest_list[i].chemical_name, source_list[i].chemical_name);
      dest_list[i].molecular_wt = source_list[i].molecular_wt;
      dest_list[i].concentration[NOCODISPOSAL]
            = source_list[i].concentration[NOCODISPOSAL];
      dest_list[i].concentration[CODISPOSAL]
            = source_list[i].concentration[CODISPOSAL];
      }

   return(count);
}


/* ========================================================================== */
/*  This function builds the Cscape type menu from the air toxics list.   */
/*  It also sizes it so it can fit in the space allowed by the other sed. */
/* ========================================================================== */

int create_air_toxics_menu(choice_menu, choice_sed, air_toxic,
                           number_air_toxics)
menu_type *choice_menu;
sed_type *choice_sed;
struct air_toxic_structure *air_toxic;
int number_air_toxics;
{
int i;
char format[80];

   *choice_menu = menu_Open();

   for (i = 0 ; i < number_air_toxics ; i++)
      {
```



```
      menu_Printf(*choice_menu, "@p[%d,0]@c[%c]@[72, ]", i, menu_color);

      sprintf(format, "@p[%d,0]@c[%c]@fp[@[30,#]]", i, menu_color);
      menu_Printf(*choice_menu, format,
                  air_toxic[i].chemical_name, &string_funcs);

      sprintf(format, "@p[%d,33]@c[%c]@fpd3[@[7,#]]", i, menu_color);
      menu_Printf(*choice_menu, format,
                  &(air_toxic[i].molecular_wt), &susdbl_funcs,
                  NULL, NULL, "%6.2lf");

      sprintf(format, "@p[%d,47]@c[%c]@fd3[@[10,#]]", i, menu_color);
      menu_Printf(*choice_menu, format,
                  &(air_toxic[i].concentration[NOCODISPOSAL]),
                  &susdbl_funcs, NULL, NULL, "%.4lf");

      sprintf(format, "@p[%d,62]@c[%c]@fd3[@[10,#]]", i, menu_color);
      menu_Printf(*choice_menu, format,
                  &(air_toxic[i].concentration[CODISPOSAL]),
                  &susdbl_funcs, NULL, NULL, "%.4lf");
      }

   menu_Flush(*choice_menu);

   *choice_sed = sed_Open(*choice_menu);

   sed_SetHeight(*choice_sed, 11);
   sed_SetWidth(*choice_sed, 72);

   sed_SetColors(*choice_sed, menu_color, menu_color, field_color);

   sed_SetPosition(*choice_sed, 9, 4);

   sed_Push(*choice_sed);

   sed_UpdateFields(*choice_sed);
}



/* ========================================================================== */
/*                This function deletes a Cscape menu.                        */
/* ========================================================================== */

int delete_air_toxics_menu(count, choice_menu, choice_sed)
int count;
menu_type choice_menu;
sed_type choice_sed;
{
   sed_Pop(choice_sed);

   menu_Close(choice_menu);
   sed_Close(choice_sed);
}


/* ========================================================================== */
/*  This function deletes a given entry (denoted by index number) from    */
/*  the air toxics list.                                                    */
/* ========================================================================== */

int delete_airtox_entry(air_toxics_ptr, entry, number_of_entries)
struct air_toxic_structure **air_toxics_ptr;
int entry, number_of_entries;
{
struct air_toxic_structure *air_toxics, *new_air_toxics;
```

```
int i, j;

/* ========================================================================== */
/*  Make a copy the the original pointer.  Then, make the original            */
/*  point to a new area of memory allocated as one air toxic entry            */
/*  smaller than before.                                                      */
/* ========================================================================== */
   air_toxics = *air_toxics_ptr;

   *air_toxics_ptr = (struct air_toxic_structure *)
      calloc((size_t) (number_of_entries - 1),
            sizeof(struct air_toxic_structure));

   if (*air_toxics_ptr == NULL)
      show_mem_error("delete_airtox_entry", "air_toxics_ptr");

/* ========================================================================== */
/*  Copy the original list to the new list, skipping the desired entry.       */
/* ========================================================================== */

   new_air_toxics = *air_toxics_ptr;

   for (j = 0, i = 0 ; i < number_of_entries ; i++)
      if (i != entry)
         {
         strcpy(new_air_toxics[j].chemical_name, air_toxics[i].chemical_name);
         new_air_toxics[j].molecular_wt = air_toxics[i].molecular_wt;
         new_air_toxics[j].concentration[NOCODISPOSAL]
               = air_toxics[i].concentration[NOCODISPOSAL];
         new_air_toxics[j].concentration[CODISPOSAL]
               = air_toxics[i].concentration[CODISPOSAL];
         j++;
         }

/* ========================================================================== */
/*                          Free the old list                                 */
/* ========================================================================== */

   free(air_toxics);

   return(number_of_entries - 1);
}


/* ========================================================================== */
/*  This function handles all the Cscape key trapping and returns the         */
/*  character received.                                                       *
/* ========================================================================== */

int accept_menu_choices(choice_sed)
sed_type choice_sed;
{
int ret;

   sed_DoFieldFkey(choice_sed, sed_GetFieldNo(choice_sed));

   ret = sed_GetBaton(choice_sed);

   sed_SetBaton(choice_sed, -1);

   return(ret);
}


/* ========================================================================== */
/*  This function activates a pop-up to receive the information on the        */
```



```
/*  new air toxic being added to the list.                                    */
/* ========================================================================== */

int add_air_toxic(chemical, mole_wt)
char *chemical;
double *mole_wt;
{
menu_type   menu;
sed_type    sed;
int         ret;

/* ========================================================================== */
/*                      Zero the string before editing.                       */
/* ========================================================================== */

   chemical[0] = '\0';

/* ========================================================================== */
/*                           Define the Cscape Menu.                          */
/* ========================================================================== */

   menu = menu_Open();

   menu_Printf(menu, "@p[0,0]@c[%c]+@[43,-]+", menu_color);
   menu_Printf(menu, "@p[1,0]¦");
   menu_Printf(menu, "@p[1,14]Add an Air Toxic");
   menu_Printf(menu, "@p[1,44]¦");
   menu_Printf(menu, "@p[2,0]¦@[43,-]¦");
   menu_Printf(menu, "@p[3,0]¦");
   menu_Printf(menu, "@p[3,44]¦");
   menu_Printf(menu, "@p[4,0]¦ Air Toxic :");
   menu_Printf(menu, "@p[4,44]¦");
   menu_Printf(menu, "@p[5,0]¦");
   menu_Printf(menu, "@p[5,44]¦");
   menu_Printf(menu, "@p[6,0]¦ Molecular Weight : ");
   menu_Printf(menu, "@p[6,44]¦");
   menu_Printf(menu, "@p[7,0]¦");
   menu_Printf(menu, "@p[7,44]¦");
   menu_Printf(menu, "@p[8,0]+@[43,-]+");
   menu_Printf(menu, "@p[4,14]@f[@[29,#]]", chemical, &string_funcs);
   menu_Printf(menu, "@p[6,21]@f[@[16,#]]", mole_wt, &susdbl_funcs);

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, menu_color, menu_color, field_color);
   sed_SetPosition(sed, 7, 17);

   sed_Push(sed);
   sed_Repaint(sed);

/* ========================================================================== */
/*        This is the main loop.  It terminates on ESC or F10.                */
/* ========================================================================== */

   do
      {
      ret = sed_Go(sed);

      if (ret) ret = verify_airtox_entry(chemical, mole_wt);
      }
   while ((ret) && (ret != FN10));

   sed_Pop(sed);

   menu_Close(menu);
```

```
   sed_Close(sed);
   return(ret);
}


int verify_airtox_entry(name, wt)
char *name;
double *wt;
{
   if (! strlen(name))
      {
      verify_error(10);
      return(1);
      }

   if (*wt <= 0.0)
      {
      verify_error(11);
      return(1);
      }

   return(FN10);
}
```

```
/* ========================================================================== */
/*                                  BUILDBOX.C                                */
/* ========================================================================== */
/*       This file contains the code for building various box types           */
/* ========================================================================== */
/*                              Created under Contract                        */
/*                                        to                                  */
/*                                EPA/AEERL/GECD/EMB                          */
/*                                                                            */
/*                                        by                                  */
/*                                                                            */
/*                                Radian Corporation                          */
/*                             Research Triangle Park, NC                     */
/* ========================================================================== */
/* ========================================================================== */
/*                                Revision History                            */
/* ========================================================================== */
/*    07/03/90   rlb       Initial Implementation                             */
/* ========================================================================== */

#include <stdio.h>
#include <cscape.h>

int buildbox(menu, color, lines, width)
menu_type menu;
int lines, width;
unsigned char color;
{
int i;

   menu_Printf(menu, "@p[0,0]@c[%c]+@[%d,-]+", color, width - 2);
   menu_Printf(menu, "@p[1,0]¦@[%d, ]¦", width - 2);
   menu_Printf(menu, "@p[2,0]¦@[%d,-]¦", width - 2);

   for (i = 3 ; i <= lines - 2 ; i++)
      menu_Printf(menu, "@p[%d,0]¦@[%d, ]¦", i, width - 2);

   menu_Printf(menu, "@p[%d,0]+@[%d,-]+", lines - 1, width - 2);

   return(0);
}


int build_scrollbox(menu, color, lines, width)
menu_type menu;
int lines, width;
unsigned char color;
{
int i;

   menu_Printf(menu, "@p[0,0]@c[%c]+@[%d,-]+", color, width - 2);
   menu_Printf(menu, "@p[1,0]¦@[%d, ]¦", width - 2);
   menu_Printf(menu, "@p[2,0]¦--@[%d,-]¦", width - 4);

   for (i = 3 ; i <= lines - 2 ; i++)
      menu_Printf(menu, "@p[%d,0]¦ ¦@[%d, ]¦", i, width - 4);

   menu_Printf(menu, "@p[%d,0]+--@[%d,-]+", lines - 1, width - 4);

   return(0);
}
```

```
/* ========================================================================== */
/*                                 CALCMAIN.C                                 */
/* ========================================================================== */
/*  This file contains the routines used to calculate the landfill gas        */
/*  emissions.                                                                */
/* ========================================================================== */
/*                           Created under Contract                           */
/*                                    to                                      */
/*                             EPA/AEERL/GECD/EMB                             */
/*                                                                            */
/*                                    by                                      */
/*                                                                            */
/*                             Radian Corporation                             */
/*                          Research Triangle Park, NC                        */
/* ========================================================================== */
/* ========================================================================== */
/*                             Revision History                               */
/* ========================================================================== */
/*    07/03/90    rlb       Initial Implementation                            */
/*    01/21/94    rlb       Changed routine to use include_air_toxics flag    */
/*                             to select codisposal option.                   */
/* ========================================================================== */

#include <stdio.h>
#include <math.h>
#include <malloc.h>
#include <cscape.h>
#include <scancode.h>
#include "landfill.h"
#include "landuser.h"
#include "defaults.h"
#include "constant.h"
#include "colors.h"

/* ========================================================================== */
/*                This is the main calculation function.                      */
/*  This function takes care of all of the results memory management as       */
/*  well as coordinating the calculation of the emission information.         */
/* ========================================================================== */

int calculation_main(landfill_info, model_info, results, done)
struct landfill_parms *landfill_info;
struct model_parms *model_info;
struct model_results *results;
boolean *done;
{
menu_type status_menu, running_menu, key_menu;
sed_type status_sed, running_sed, key_sed;
int i, index, j, year;
double ch4_rate;
double model_function();

double mass_to_volume();
double volume_to_mass();

/* ========================================================================== */
/*  Start out by setting the flag indicating whether the calculation was      */
/*  done or not to FALSE.  This will not get reset until the calculation      */
/*  is over.                                                                  */
/* ========================================================================== */

   *done = FALSE;

/* ========================================================================== */
/*  Build the status box.  This box will hold notices for the user on the     */
/*  action currently being taken by the computer.  Initialize the year        */
```



```
/*  variable at the first year being calculated.  This will be changed       */
/*  and displayed to let the user know how far the calculation has            */
/*  progressed                                                                */
/* ========================================================================== */

   build_status(&status_sed, &status_menu);

   year = landfill_info->year_opened + 1;

/* ========================================================================== */
/*  This is the major memory management section.  First, clear any data       */
/*  left over from a previous run.  Then, make new space the length of        */
/*  new calculation for each of the tracked variables.                        */
/*                                                                            */
/*  There is an inherent advantage to callocing array space a bit at a        */
/*  time - you can asked for memory spaning more than one segment total.      */
/* ========================================================================== */

   clear_previous_results(results);

   results->year_count = landfill_info->year_closed -
         landfill_info->year_opened + config.years_passed_closure;

   results->air_toxics_count = model_info->number_air_toxics;

   results->year_list = (int *) calloc((size_t) results->year_count,
                                       sizeof(int));

   if (results->year_list == NULL)
      {
      clear_previous_results(results);
      show_mem_error2();
      destroy_status(&status_sed, &status_menu);
      return(-1);
      }

   results->annual_ch4 = (double *) calloc((size_t) results->year_count,
      sizeof(double));
   if (results->annual_ch4 == NULL)
      {
      clear_previous_results(results);
      show_mem_error2();
      destroy_status(&status_sed, &status_menu);
      return(-1);
      }

   results->annual_co2 = (double *) calloc((size_t) results->year_count,
                                           sizeof(double));
   if (results->annual_co2 == NULL)
      {
      clear_previous_results(results);
      show_mem_error2();
      destroy_status(&status_sed, &status_menu);
      return(-1);
      }

   results->annual_nmoc = (double *) calloc((size_t) results->year_count,
                                            sizeof(double));
   if (results->annual_nmoc == NULL)
      {
      clear_previous_results(results);
      show_mem_error2();
      destroy_status(&status_sed, &status_menu);
      return(-1);
      }
```

```
/* ========================================================================== */
/*  Only make space for air toxics if they are included in the calc.          */
/*                                                                            */
/*  The arrangement of memory for the air toxic emission is an array          */
/*  of pointers for each year of the calculation pointing to an array of      */
/*  doubles containing the emissions for each air toxic calculated.           */
/* ========================================================================== */

   if (model_info->include_air_toxics != NO_AT)
      {
      results->annual_air_toxics = (double **)
         calloc((size_t) model_info->number_air_toxics, sizeof(double *));
      if (results->annual_air_toxics == NULL)
         {
         clear_previous_results(results);
         show_mem_error2();
         destroy_status(&status_sed, &status_menu);
         return(-1);
         }

      for (i = 0 ; i < model_info->number_air_toxics ; i++)
         {
         results->annual_air_toxics[i] = (double *)
            calloc((size_t) results->year_count, sizeof(double));

         if (results->annual_air_toxics[i] == NULL)
            {
            clear_previous_results(results);
            show_mem_error2();
            destroy_status(&status_sed, &status_menu);
            return(-1);
            }
         }
      }

/* ========================================================================== */
/*  Allocate space for the refuse acceptance history of the landfill.         */
/*  This array holds acceptance rate for each year until the last year        */
/*  the calculation is done.                                                  */
/* ========================================================================== */

   results->refuse_data = (double *) calloc((size_t) results->year_count,
                  sizeof(double));

   if (results->refuse_data == NULL)
      {
      clear_previous_results(results);
      show_mem_error2();
      destroy_status(&status_sed, &status_menu);
      return(-1);
      }

/* ========================================================================== */
/*  Fill the acceptance rate list with the past history information and       */
/*  project forward until the last calculation year.                          */
/* ========================================================================== */

   get_refuse_list(landfill_info, results->refuse_data);

/* ========================================================================== */
/*  If we have gotten this far, the memory allocation steps worked and we     */
/*  are ready to do the calculation.  Pop-up a box to tell the user which     */
/*  year we are currently calculating for.                                    */
/* ========================================================================== */

   build_running(&running_sed, &running_menu, &year);
```



```
   show_key_help(&key_menu, &key_sed, "ESC = Interrupt");

/* ========================================================================== */
/*  Set a flag that tells if the calculation was completed without            */
/*  user interruption.                                                        */
/* ========================================================================== */

   *done = TRUE;

/* ========================================================================== */
/*                              Main Loop                                     */
/*  This loop calculates the methane, carbon dioxide, nmoc, and air toxic     */
/*  emissions (if any) from the year after landfill start until the last      */
/*  desired calculation year (10 years past closure).                         */
/* ========================================================================== */

   for (i = 0 ; i < results->year_count ; i++, year++)
      {
      results->year_list[i] = year;
/* ========================================================================== */
/*        Update the status box with the current calculation year             */
/* ========================================================================== */

      sed_RepaintFields(running_sed);

/* ========================================================================== */
/*             Get the current methane generation rate (Cft/yr)               */
/* ========================================================================== */
      ch4_rate = model_function(model_info->k, model_info->l0,
                                results->refuse_data, i);

      results->annual_ch4[i] = ch4_rate;

/* ========================================================================== */
/*  The carbon dioxide generation rate is assumed to be the balance of        */
/*  landfill gas after accounting for the methane (Cft/yr).                   */
/* ========================================================================== */

      results->annual_co2[i] = ch4_rate * model_info->co2
                             / model_info->ch4;

/* ========================================================================== */
/*  NMOC (non-methane organic compounds) are assumed to be a negligible       */
/*  constituent of landfill gas.  The emission is expected to be directly     */
/*  proportional to the landfill gas generation rate (the 1E6 compensates     */
/*  for the NMOC entry units of ppmv.  Units in Cft/yr.                       */
/* ========================================================================== */

      results->annual_nmoc[i] = ch4_rate * 100.0 * model_info->nmoc
                                / (1.0E6 * model_info->ch4);

/* ========================================================================== */
/*                     Convert saved results into Mg/yr                       */
/* ========================================================================== */

      results->annual_nmoc[i] =
         volume_to_mass(results->annual_nmoc[i] * FT3_TO_M3,
                        DEFAULT_NMOC_MOLE_WT, DEFAULT_TEMP,
                        DEFAULT_PRESSURE) / 1000.0;

/* ========================================================================== */
/*  If air toxic analysis is selected, calculate the emission using the       */
/*  the same assumption as in the NMOC case.                                  */
/* ========================================================================== */

      if (model_info->include_air_toxics != NO_AT)
```

```
         for (j = 0 ; j < model_info->number_air_toxics ; j++)
            results->annual_air_toxics[j][i] =
                ch4_rate * 100.0
                  * model_info->air_toxics[j].concentration[model_info->include_
air_toxics] /
                  (1.0E6 * model_info->ch4);

/* ================================================================== */
/*  Watch the keyboard.  If the user presses ESC, he wants to abort the  */
/*  calculation.  Make sure to set the done flag to FALSE to prevent     */
/*  displaying of the incomplete calculation set.                        */
/* ================================================================== */

      if (kb_Check())
         if (kb_Read() == ESC)
            {
            *done = FALSE;
            break;
            }
      }

/* ================================================================== */
/*                              End Loop                              */
/* ================================================================== */

/* ================================================================== */
/*                        Get rid of the status box                   */
/* ================================================================== */

   destroy_key_help(key_menu, key_sed);
   destroy_running(&running_sed, &running_menu);
   destroy_status(&status_sed, &status_menu);

/* ================================================================== */
/*  If the calculation was done completely, analyze the data and produce */
/*  an executive summary of the results on the screen.                   */
/* ================================================================== */

   if (*done) display_summary(landfill_info, results);
}



/* ================================================================== */
/*  This function performs the dirty work of cleaning up the memory left */
/*  over from a previous model run.                                      */
/* ================================================================== */

int clear_previous_results(results)
struct model_results *results;
{
int i;

   if (results->year_list != NULL) free(results->year_list);
   if (results->refuse_data != NULL) free(results->refuse_data);

   if (results->annual_ch4 != NULL) free(results->annual_ch4);
   if (results->annual_co2 != NULL) free(results->annual_co2);
   if (results->annual_nmoc != NULL) free(results->annual_nmoc);

/* ================================================================== */
/*  Cleaning up the air toxics array requires individually results for   */
/*  each one of the air toxics in the study.                             */
/* ================================================================== */
```



```
   for (i = 0 ; i < results->air_toxics_count ; i++)
      if (results->annual_air_toxics[i] != NULL)
         free(results->annual_air_toxics[i]);

   if (results->annual_air_toxics != NULL) free(results->annual_air_toxics);

   results->year_count = 0;
   results->air_toxics_count = 0;

   results->year_list = NULL;
   results->refuse_data = NULL;

   results->annual_ch4 = NULL;
   results->annual_co2 = NULL;
   results->annual_nmoc = NULL;
   results->annual_air_toxics = NULL;

   return(0);
}


/* ================================================================== */
/*  These functions use the ideal gas law to convert mass to volume and  */
/*                          volume to mass.                              */
/*      All parameters are assumed to be in the correct units (MKS).     */
/* ================================================================== */


double mass_to_volume(mass, mole_wt, temp, pressure)
double mass, mole_wt, temp, pressure;
{
   return(mass * (1000.0 * R) * temp / (mole_wt * pressure));
}



double volume_to_mass(volume, mole_wt, temp, pressure)
double volume, mole_wt, temp, pressure;
{
   return((mole_wt * pressure * volume) / (1000.0 * R * temp));
}


/* ================================================================== */
/*  This is the actual model function.   It attempts to model the methane */
/*  emissions by assuming the generation is being emitted exponentially  */
/*  at different positions allow the emission curve by each subset of    */
/*  waste materials, depending on the time between the current year and  */
/*  the initial placement of the waste subset.  The emissions for each   */
/*  submass is calculated and summed to determine the total emission of  */
/*  the landfill as a whole.                                             */
/*                                                                       */
/*  Table of Variables                                                   */
/*                                                                       */
/*  k       The methane generation decay rate constant.                  */
/*  L0      The methane generation capacity constant.                    */
/*  refuse  The refuse acceptance rate list starting from the opening    */
/*          to the end calc year.                                        */
/*  i       The current year index (0 = 1 year after opening)            */
/* ================================================================== */

double model_function(k, l0, refuse, i)
double l0, k, *refuse;
int i;
{
int j;
```

```
double sum = 0;

   for (j = 0 ; j <= i ; j++)
      sum += refuse[j] * exp(-k * (i - j));

   return(k * 10 * sum);
}


/* ========================================================================== */
/*  This function pieces together the landfill refuse acceptance rate       */
/*  list through the last calculation day using the user entered refuse     */
/*  in place list and the closure data.                                     */
/* ========================================================================== */

int get_refuse_list(data, refuse)
struct landfill_parms *data;
double *refuse;
{
int i, j, index_closed;
double sum = 0;

   index_closed = data->year_closed - data->year_opened - 1;

   sum += refuse[0] = data->refuse_in_place[0];

/* ========================================================================== */
/*  Starting with the refuse in place list, get the difference between      */
/*  successive years and place that in the acceptance rate list.  Keep      */
/*  doing this until we reach the current year or the closure year.         */
/* ========================================================================== */

   for (i = 1 ; (i < data->number_of_periods) && (i < index_closed) ; i++)
      {
      refuse[i] = data->refuse_in_place[i] - data->refuse_in_place[i - 1];
      sum += refuse[i];
      }

/* ========================================================================== */
/*  For the balance of the list, fill it in with the last rate necessary    */
/*  to close by closure year (determine in the periods entry process).      */
/* ========================================================================== */

   for (j = 0; j < (data->year_closed - data->current_year - 1) ; j++, i++)
      {
      refuse[i] = data->last_rate;
      sum += refuse[i];
      }

/* ========================================================================== */
/*  Put the balance of the refuse required to reach capacity in the         */
/*  closure cell.  (This step was done because the original method of       */
/*  calculating the acceptance rate break down was based on the last        */
/*  observed acceptance rate continuing until closure.  Consequently, the   */
/*  balance of the list remaining just before closure may be smaller than   */
/*  the actual last rate observed and needed to be calculated.  By          */
/*  leaving this step, it will compensate for changes in the closure        */
/*  algorithm.)                                                             */
/* ========================================================================== */

   if (index_closed > 1)
     refuse[i++] = data->capacity - sum;

/* ========================================================================== */
/*  Fill in the remaining part of the list with zeros (no waste accepted)   */
/* ========================================================================== */
```



```
   for (j = 0 ; j < config.years_passed_closure ; j++, i++)
      refuse[i] = 0.0;
}


/* ========================================================================== */
/*  This function actually builds the status box.  It originally contains  */
/*  a memory allocation message.  This gets overwritten by the year         */
/*  later.                                                                  */
/* ========================================================================== */

int build_status(sed, menu)
menu_type   *menu;
sed_type    *sed;
{

   *menu = menu_Open();

   menu_Printf(*menu, "@p[0,0]@c[%c]+@[36,-]+", info_color);
   menu_Printf(*menu, "@p[1,0]|");
   menu_Printf(*menu, "@p[1,8]Calculation in Progress");
   menu_Printf(*menu, "@p[1,37]|");
   menu_Printf(*menu, "@p[2,0]|@[36,-]|");
   menu_Printf(*menu, "@p[3,0]|");
   menu_Printf(*menu, "@p[3,37]|");
   menu_Printf(*menu, "@p[4,0]|");
   menu_Printf(*menu, "@p[4,9]Allocating Memory");
   menu_Printf(*menu, "@p[4,37]|");
   menu_Printf(*menu, "@p[5,0]|");
   menu_Printf(*menu, "@p[5,37]|");
   menu_Printf(*menu, "@p[6,0]|");
   menu_Printf(*menu, "@p[6,10]Please Wait ...");
   menu_Printf(*menu, "@p[6,37]|");
   menu_Printf(*menu, "@p[7,0]|");
   menu_Printf(*menu, "@p[7,37]|");
   menu_Printf(*menu, "@p[8,0]+@[36,-]+");

   menu_Flush(*menu);

   *sed = sed_Open(*menu);
   sed_SetColors(*sed, info_color, info_color, info_color);
   sed_SetPosition(*sed, 7, 20);

   sed_Push(*sed);
   sed_Repaint(*sed);

   return(0);
}


/* ========================================================================== */
/*  This function builds the year status display information.  It is        */
/*  overlayed on the status box.                                            */
/* ========================================================================== */

int build_running(sed, menu, year)
menu_type   *menu;
sed_type    *sed;
int *year;
{
int i;

   *menu = menu_Open();
```

```
   for (i = 0 ; i < 5 ; i++)
      menu_Printf(*menu, "@p[%d,0]@c[%c]@[35, ]", i, info_color);

   menu_Printf(*menu, "@p[1,12]@c[%c]Current Year", info_color);
   menu_Printf(*menu, "@p[3,16]@f[####]",
     year, &susint_funcs);

   menu_Flush(*menu);

   *sed = sed_Open(*menu);
   sed_SetColors(*sed, info_color, info_color, info_color);
   sed_SetPosition(*sed, 10, 21);

   sed_SetWidth(*sed, 35);
   sed_SetHeight(*sed, 5);

   sed_Repaint(*sed);

   return(0);
}

/* ================================================================== */
/*   The following two functions destroy the menu items created in the */
/*   above functions.                                                  */
/* ================================================================== */


int destroy_running(sed, menu)
sed_type *sed;
menu_type *menu;
{
   menu_Close(*menu);
   sed_Close(*sed);
   return(0);
}


int destroy_status(sed, menu)
sed_type *sed;
menu_type *menu;
{
   sed_Pop(*sed);

   menu_Close(*menu);
   sed_Close(*sed);
   return(0);
}
```

```
/* ================================================================== */
/*                              CHEMCOMP.C                            */
/* ================================================================== */
/*  This file contains the code for displaying and altering the chemical  */
/*  composition of landfill gas (not including individual air toxics).    */
/* ================================================================== */
/*                         Created under Contract                     */
/*                                  to                                */
/*                           EPA/AEERL/GECD/EMB                       */
/*                                                                    */
/*                                  by                                */
/*                                                                    */
/*                           Radian Corporation                       */
/*                        Research Triangle Park, NC                  */
/* ================================================================== */
/* ================================================================== */
/*                           Revision History                         */
/* ================================================================== */
/*    07/03/90   rlb      Initial Implementation                      */
/*    02/08/94   rlb      Modified to permit default NMOC selections. */
/*    04/14/94   rlb      Modified to change screen order to better   */
/*                          support the NMOC selection.  Added new    */
/*                          warning screen.                           */
/* ================================================================== */

#include <stdio.h>
#include <cscape.h>
#include <scancode.h>
#include <string.h>
#include "landfill.h"
#include "defaults.h"
#include "landuser.h"
#include "helpasgn.h"
#include "colors.h"

#define YES  1
#define NO   0

/* ================================================================== */
/*              Declare the choices for this selection                */
/* ================================================================== */

static char *include_ptrs[] = {"No Codisposal", "Codisposal", "None",
                               NULL};


/* ================================================================== */
/*     This is the main menuing function for edit chemical composition    */
/* ================================================================== */

int edit_chem_comp(model_info)
struct model_parms *model_info;
{
void (*old_fkey)();
struct model_parms scratch_model_parms;
menu_type   menu, key_menu;
sed_type    sed, key_sed;
int         ret, old_include_at;
struct select_struct air_toxic_choice;

int copy_model_parms(struct model_parms *, struct model_parms *);
void recalc_fkey();

   reset_insert_status();

   old_fkey = susdbl_funcs.fkey;
```



```
   susdbl_funcs.fkey = recalc_fkey;

/* ================================================================== */
/*        Copy the original model parameters into a scratch area.     */
/* ================================================================== */

   copy_model_parms(&scratch_model_parms, model_info);

/* ================================================================== */
/*       Define the contents of the Cscape toggle field function.     */
/* ================================================================== */

   old_include_at = air_toxic_choice.number = scratch_model_parms.include_air_to
xics;
   air_toxic_choice.list = include_ptrs;

/* ================================================================== */
/*                          Define the Cscape Menu                    */
/* ================================================================== */

   menu = menu_Open();

   buildbox(menu, menu_color, 15, 48);
   menu_Printf(menu, "@p[1,11]Edit Chemical Composition");
   menu_Printf(menu, "@p[4,5]Air Pollutants Option:");
   menu_Printf(menu, "@p[6,5]CH   :");
   menu_Printf(menu, "@p[6,32]%% Volume");
   menu_Printf(menu, "@p[7,7]4");
   menu_Printf(menu, "@p[9,5]CO   :");
   menu_Printf(menu, "@p[9,32]%% Volume");
   menu_Printf(menu, "@p[10,7]2");
   menu_Printf(menu, "@p[12,5]NMOC :");
   menu_Printf(menu, "@p[12,32]ppm Volume");
   menu_Printf(menu, "@p[4,29]@f[@[15,#]]",
     &air_toxic_choice, &myselect_funcs);
   menu_Printf(menu, "@p[6,12]@f[@[18,#]]",
     &(scratch_model_parms.ch4), &susdbl_funcs);
   menu_Printf(menu, "@p[9,12]@fp[@[18,#]]",
     &(scratch_model_parms.co2), &susdbl_funcs);
   menu_Printf(menu, "@p[12,12]@f[@[18,#]]",
     &(scratch_model_parms.nmoc), &susdbl_funcs);

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, menu_color, menu_color, field_color);
   sed_SetPosition(sed, 5, 16);

   sed_Push(sed);
   sed_Repaint(sed);
   show_key_help(&key_menu, &key_sed, chemcomp_help_string);

/* ================================================================== */
/*                          The Main Control Loop                     */
/*                   The loop terminates on ESC or F10.               */
/* ================================================================== */

   do
      {

/* ================================================================== */
/*   Accept field entry using Cscapes routine.  Since the special keys    */
/*  have been redefined for this program, if exit occurs, check the exit  */
/*  key to determine if the exit was caused by a special key.             */
/* ================================================================== */
```

```
      ret = sed_Go(sed);

      scratch_model_parms.include_air_toxics
                  = select_number(air_toxic_choice);

      if (old_include_at != scratch_model_parms.include_air_toxics)
         {
         warn_about_nmoc_default();
         old_include_at = scratch_model_parms.include_air_toxics;
         }

      if (ret)
         {
         if (verify_chem_comp(scratch_model_parms) == FN10)
            {
            scratch_model_parms.co2 = 100.0 - scratch_model_parms.ch4;

            sed_RepaintFields(sed);
            }
         else
            sed_GotoField(sed, 1);
         }


      switch (ret)
         {
/* ================================================================== */
/*  F1 - Help request.                                                */
/*                                                                    */
/*  Call the centralized help function (my facility - not Cscape's).  */
/* ================================================================== */
         case FN1:
            compile_help(EDIT_CHEM_COMP);
            sed_SetBaton(sed, -1);
            break;

/* ================================================================== */
/*  F2 - Reset values to their default settings.                      */
/* ================================================================== */
         case FN2:
            scratch_model_parms.include_air_toxics
                     = select_number(air_toxic_choice);
            reset_defaults(&scratch_model_parms, sed_GetFieldNo(sed));

            sed_RepaintFields(sed);
            break;
         }

/* ================================================================== */
/*  If they are trying to accept entry with an F10, verify the data first */
/*  before allowing them to leave.  If a value is bad, redefine ret to    */
/*  another value before return.  If all values are OK, return an F10 and */
/*  let the user exit.                                                */
/* ================================================================== */

      if (ret == FN10) ret = verify_chem_comp(scratch_model_parms);
      }
   while ((ret) && (ret != FN10));

/* ================================================================== */
/*                          End of Main Loop                          */
/* ================================================================== */

/* ================================================================== */
/*  Get their choice for inclusion of air toxics and write it into the    */
/*  scratch area.                                                     */
```



```
/* ================================================================== */

   scratch_model_parms.include_air_toxics
                  = select_number(air_toxic_choice);

/* ================================================================== */
/*  If the accept key was pressed, the user accepted the data and it was  */
/*  verified.  Copy the data into the original structure.             */
/* ================================================================== */
   if (ret == FN10)
      copy_model_parms(model_info, &scratch_model_parms);

/* ================================================================== */
/*                  Done now.  Destroy the menu and exit              */
/* ================================================================== */
   destroy_key_help(key_menu, key_sed);
   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);

   susdbl_funcs.fkey = old_fkey;

   return(ret);
}


/* ================================================================== */
/*      This function copies of the model parm structures to another  */
/* ================================================================== */

int copy_model_parms(dest, source)
struct model_parms *dest, *source;
{
   dest->l0 = source->l0;
   dest->k = source->k;
   dest->nmoc = source->nmoc;
   dest->ch4 = source->ch4;
   dest->co2 = source->co2;
   dest->include_air_toxics = source->include_air_toxics;
   dest->air_toxics = source->air_toxics;
   dest->number_air_toxics = source->number_air_toxics;
}


/* ================================================================== */
/*  This function resets the requested value to its default state.  Most  */
/*  the value passed in is a Cscape field number which identifies the     */
/*  parameter to be changed.                                          */
/* ================================================================== */

int reset_defaults(data, number)
struct model_parms *data;
int number;
{
   switch (number)
      {
      case 3:
         if (config.default_mode == AP_MODE)
            {
            if (data->include_air_toxics == CODISPOSAL)
               data->nmoc = DEFAULT_AP42_NMOC_CODIS;
            else data->nmoc = DEFAULT_AP42_NMOC_NOCO;
            }
         else data->nmoc = DEFAULT_REGULATORY_NMOC;
```

```
         break;

      case 1:
         data->ch4 = DEFAULT_CH4;
         break;

      case 2:
         data->co2 = DEFAULT_CO2;
         break;
      }
}

/* ========================================================================== */
/*  This function checks the data for inconsistencies.  If the value is   */
/*  incorrect, a box informs the user of the error and this function      */
/*  returns a value other than F10, the accepting code.                   */
/* ========================================================================== */

int verify_chem_comp(data)
struct model_parms data;
{
/* ========================================================================== */
/*  Do the sums of the percentages of methane and carbon dioxide yield a  */
/*  result greater than one?                                              */
/* ========================================================================== */

   if (data.ch4 > 100.0)
      {
      verify_error(7);
      return(1);
      }

/* ========================================================================== */
/*         Is the methane percentage equal to (or less than) zero?        */
/*  Special Note:  this model assumes the landfill produces methane and   */
/*  cannot be used if it does not.                                        */
/* ========================================================================== */
   if (data.ch4 <= 0.0)
      {
      verify_error(9);
      return(1);
      }

   if (data.nmoc > 1E6)
      {
      verify_error(13);
      return 1;
      }

/* ========================================================================== */
/*               Data checks!  Return the accept code.                    */
/* ========================================================================== */
   return(FN10);
}
```

```
/* ========================================================================== */
/*                                  CLOSURE.C                                 */
/* ========================================================================== */
/*  This file contains the code for displaying and altering the landfill      */
/*  closure date determined from the period data.                             */
/* ========================================================================== */
/*                            Created under Contract                          */
/*                                    to                                      */
/*                              EPA/AEERL/GECD/EMB                            */
/*                                                                            */
/*                                    by                                      */
/*                                                                            */
/*                              Radian Corporation                            */
/*                           Research Triangle Park, NC                       */
/* ========================================================================== */
/* ========================================================================== */
/*                              Revision History                              */
/* ========================================================================== */
/*    07/03/90   rlb       Initial Implementation                             */
/* ========================================================================== */

#include <stdio.h>
#include <cscape.h>
#include <scancode.h>
#include "landfill.h"
#include "landuser.h"
#include "defaults.h"
#include "helpasgn.h"
#include "colors.h"

int display_closure_year(data)
struct landfill_parms *data;
{
menu_type   menu, key_menu;
sed_type    sed, key_sed;
int         ret;
double scratch_accept_rate;
int scratch_year_closed;

/* ========================================================================== */
/*  Make a copy of the landfill closure parameters so we can change them      */
/*  without any problems.                                                     */
/* ========================================================================== */

   scratch_accept_rate = data->last_rate;
   scratch_year_closed = data->year_closed;

/* ========================================================================== */
/*                          Define the Cscape Menu                            */
/* ========================================================================== */

   menu = menu_Open();

   menu_Printf(menu, "@p[0,0]@c[%c]+@[44,-]+", menu_color);
   menu_Printf(menu, "@p[1,0]|");
   menu_Printf(menu, "@p[1,45]|");
   menu_Printf(menu, "@p[2,0]|");
   menu_Printf(menu, "@p[2,9]For the period data entered,");
   menu_Printf(menu, "@p[2,45]|");
   menu_Printf(menu, "@p[3,0]| the expected year of closure is determined |");
   menu_Printf(menu, "@p[4,0]|");
   menu_Printf(menu, "@p[4,20]to be");
   menu_Printf(menu, "@p[4,45]|");
   menu_Printf(menu, "@p[5,0]|");
   menu_Printf(menu, "@p[5,45]|");
   menu_Printf(menu, "@p[6,0]|");
```



```
   menu_Printf(menu, "@p[6,45]|");
   menu_Printf(menu, "@p[7,0]|");
   menu_Printf(menu, "@p[7,45]|");
   menu_Printf(menu, "@p[8,0]|    assuming a refuse acceptance rate of    |");
   menu_Printf(menu, "@p[9,0]|");
   menu_Printf(menu, "@p[9,45]|");
   menu_Printf(menu, "@p[10,0]|");
   menu_Printf(menu, "@p[10,29]Mg/year");
   menu_Printf(menu, "@p[10,45]|");
   menu_Printf(menu, "@p[11,0]|");
   menu_Printf(menu, "@p[11,45]|");
   menu_Printf(menu, "@p[12,0]|     from current year to closure year.     |");
   menu_Printf(menu, "@p[13,0]|");
   menu_Printf(menu, "@p[13,45]|");
   menu_Printf(menu, "@p[14,0]+@[44,-]+");
   menu_Printf(menu, "@p[6,20]@f[####]",
     &scratch_year_closed, &susint_funcs);
   menu_Printf(menu, "@p[10,13]@fp[@[14,#]]",
     &scratch_accept_rate, &susdbl_funcs);

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, menu_color, menu_color, field_color);
   sed_SetPosition(sed, 5, 17);

   sed_Push(sed);

   sed_Repaint(sed);

/* ========================================================================== */
/*  If the closure year has already been encountered, the screen is info      */
/*  only and cannot be changed so it doesn't need help.                       */
/* ========================================================================== */

   if (data->refuse_in_place[data->number_of_periods - 1] < data->capacity)
      show_key_help(&key_menu, &key_sed, generic_help_string);
   else
      show_key_help(&key_menu, &key_sed, continue_help_string);

/* ========================================================================== */
/*  If the closure date has yet to be encountered, allow the user to edit     */
/*  the computer generated value.  First, the closure date was already        */
/*  determined from periods.c.  Recalculate the last rate using the           */
/*  closure date.  This has the affect of making the last rate an exact       */
/*  average rate to reach closure.  Then, activate the field entry loop.      */
/* ========================================================================== */

   if (data->refuse_in_place[data->number_of_periods - 1] < data->capacity)
      {
      calculate_acceptance_rate(data, scratch_year_closed,
            &scratch_accept_rate);

      sed_RepaintFields(sed);

      do
         {
         ret = sed_Go(sed);

         switch (ret)
            {
/* ========================================================================== */
/*  F1 - Help requested                                                       */
/*                                                                            */
/*  Call the centralized help facility.                                       */
/* ========================================================================== */
```

```
            case FN1:
               compile_help(DISPLAY_CLOSURE_YEAR);
               sed_SetBaton(sed, -1);
               break;

/* ========================================================================= */
/*  F2 - Reset the closure year to its default value                         */
/* ========================================================================= */
            case FN2:
               scratch_year_closed = data->year_closed;
               break;
            }

/* ========================================================================= */
/*  If ret is non-zero, check to make sure the closure date listed is a      */
/*  valid entry.  If so, recalculate the acceptance rate assuming the new    */
/*  closure date and redisplay.                                              */
/* ========================================================================= */

         if (ret)
            {

#ifdef DEBUG
            printf("Validating closure data\n");
            getchar();
#endif

            if (! validate_closure_data(data, &scratch_year_closed))
               {
               calculate_acceptance_rate(data, scratch_year_closed,
                                         &scratch_accept_rate);

               sed_RepaintFields(sed);
               }
            else
               ret = -1;
            }
         }
      while ((ret) && (ret != FN10));

/* ========================================================================= */
/*  Since the data was verified on the fly and F10 was pressed, the user     */
/*  has accepted the closure date displayed.  Update the original struct.   */
/* ========================================================================= */

      if (ret == FN10)
         {
         data->year_closed = scratch_year_closed;
         data->last_rate = scratch_accept_rate;
         }
      }

/* ========================================================================= */
/*                       End editable closure year case                      */
/* ========================================================================= */

   else

/* ========================================================================= */
/*  The closure date has already been encountered so the user shouldn't      */
/*  change the closure date.  Just watch for a key and terminate.            */
/* ========================================================================= */

      kb_Read();

   destroy_key_help(key_menu, key_sed);
```



```
   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);
   return(ret);
}


/* ========================================================================= */
/*  This function checks to see if the date entered is a legitimate value   */
/* ========================================================================= */

int validate_closure_data(data, year)
struct landfill_parms *data;
int *year;
{
/* ========================================================================= */
/*         Is the closure year greater than the current year?                */
/*  Since the calling function goes another path if the closure year has     */
/*  been encountered, this is not a legitimate entry.                        */
/* ========================================================================= */

   if (data->current_year >= *year)
      {
      verify_error(5);
      return(1);
      }

/* ========================================================================= */
/*           Does the year exceed the program design limits?                 */
/* ========================================================================= */

   if (*year - data->year_opened > MAX_DURATION)
      {
      verify_error(6);
      return(1);
      }

/* ========================================================================= */
/*                            Year OK - return zero                          */
/* ========================================================================= */

   return(0);
}


/* ========================================================================= */
/*  This function calculates the acceptance rate given the closure year.     */
/*  This is a two step process originating with periods.c.  First, the       */
/*  closure year is calculated based on the last observed (or logically      */
/*  interpolated) acceptance rate.  The capacity is reduced by whole         */
/*  numbers of last rates until the remainder is less than the last rate.    */
/*  The closure year is therefore the whole number of last rates required    */
/*  with one more added if the remainder is non-zero.  Consequently, the     */
/*  last rate is not a uniform value over the years and makes computing      */
/*  the last rate displayed in the closure screen inexact.  This function    */
/*  is a logical companion to the one in periods and it rationalizes the     */
/*  two rates.                                                               */
/* ========================================================================= */

int calculate_acceptance_rate(data, close, rate)
struct landfill_parms *data;
int close;
double *rate;
{
int remaining, i;
```

```
/* ========================================================================== */
/*  First, determining the number of remaining years and the remaining       */
/*  refuse until capacity is reached.  Calculate the acceptance rate and     */
/*  return it.                                                               */
/* ========================================================================== */

   remaining = close - data->current_year;
   i = data->number_of_periods - 1;

   if (remaining <= 0)
      {
      *rate = 0.0;
      return(0);
      }

   *rate = (data->capacity - data->refuse_in_place[i]) / remaining;
}
```

```
/* ========================================================================== */
/*                                  CONFIG.C                                  */
/* ========================================================================== */
/*  This file contains the code for displaying and altering the landfill      */
/*  program configuration settings.                                           */
/* ========================================================================== */
/*                            Created under Contract                          */
/*                                      to                                    */
/*                              EPA/AEERL/GECD/EMB                            */
/*                                                                            */
/*                                      by                                    */
/*                                                                            */
/*                              Radian Corporation                            */
/*                           Research Triangle Park, NC                       */
/* ========================================================================== */
/* ========================================================================== */
/*                             Revision History                               */
/* ========================================================================== */
/*    07/03/90   rlb       Initial Implementation                             */
/* ========================================================================== */

#include <stdio.h>
#include <cscape.h>
#include <scancode.h>
#include <string.h>
#include "landfill.h"
#include "landuser.h"
#include "defaults.h"
#include "helpasgn.h"
#include "colors.h"

/* ========================================================================== */
/*  This function handles provides the menu for changing and writing the      */
/*  landfill program configuration settings to disk.                          */
/* ========================================================================== */

/* -------------------------------------------------------------------------- */
/*  First, define the device settings for the toggle selection function.      */
/* -------------------------------------------------------------------------- */

static char *devices[] = {"LPT1", "LPT2", "COM1", "COM2", NULL};
static char *defaults[] = {"REGULATORY", "AP-42", NULL};

int configure_system()
{
FILE *config_file;
menu_type   menu, key_menu;
sed_type    sed, key_sed;
struct select_struct device_choice;
struct select_struct defaults_choice;
int ret, len, i;
int scratch_limit;
char scratch_dir[265];
int verify_ypc(int years);

/* ---------------------------------------------------------------------- */
/* Set the initial value of the scratch years passed closure.             */
/* ---------------------------------------------------------------------- */

   scratch_limit = config.years_passed_closure;


/* ========================================================================== */
/*                          Build the device choices.                         */
/* ========================================================================== */
```



```
   device_choice.number = config.print_device;
   device_choice.list = devices;

/* ========================================================================== */
/*                          Build the defaults choices.                       */
/* ========================================================================== */


   defaults_choice.number = config.default_mode;
   defaults_choice.list = defaults;


/* ========================================================================== */
/*              Build the working directory ticker string.                    */
/* ========================================================================== */

   tick_SetLen(scratch_dir, 254);
   tick_SetOffset(scratch_dir, 0);

   len = strlen(config.working_dir);
   if (*(config.working_dir + len - 1) != '\\')
      {
      *(config.working_dir + len) = '\\';
      *(config.working_dir + len + 1) = '\0';
      }

   strcpy(tick_GetString(scratch_dir), config.working_dir);

/* ========================================================================== */
/*                            Define the Cscape Menu                          */
/* ========================================================================== */

   reset_insert_status();

   menu = menu_Open();

   menu_Printf(menu, "@p[0,0]@c[%c]+@[48,-]+", menu_color);
   menu_Printf(menu, "@p[1,0]¦");
   menu_Printf(menu, "@p[1,13]Edit System Configuration");
   menu_Printf(menu, "@p[1,49]¦");
   menu_Printf(menu, "@p[2,0]¦@[48,-]¦");
   menu_Printf(menu, "@p[3,0]¦");
   menu_Printf(menu, "@p[3,49]¦");
   menu_Printf(menu, "@p[4,0]¦ Working Directory :");
   menu_Printf(menu, "@p[4,49]¦");
   menu_Printf(menu, "@p[5,0]¦");
   menu_Printf(menu, "@p[5,49]¦");
   menu_Printf(menu, "@p[6,0]¦ Default Print Device :");
   menu_Printf(menu, "@p[6,49]¦");
   menu_Printf(menu, "@p[7,0]¦");
   menu_Printf(menu, "@p[7,49]¦");
   menu_Printf(menu, "@p[8,0]¦ Defaults Mode :");
   menu_Printf(menu, "@p[8,49]¦");
   menu_Printf(menu, "@p[9,0]¦");
   menu_Printf(menu, "@p[9,49]¦");
   menu_Printf(menu, "@p[10,0]¦ Calculation Years Passed Closure: ");
   menu_Printf(menu, "@p[10,49]¦");
   menu_Printf(menu, "@p[11,0]¦");
   menu_Printf(menu, "@p[11,49]¦");
   menu_Printf(menu, "@p[12,0]+@[48,-]+");
   menu_Printf(menu, "@p[4,22]@f[@[26,#]]", scratch_dir, &ticker_funcs);
   menu_Printf(menu, "@p[6,25]@f[######]", &device_choice, &select_funcs);
   menu_Printf(menu, "@p[8,18]@f[@[11,#]]", &defaults_choice, &select_funcs);
   menu_Printf(menu, "@p[10,37]@f[###]", &scratch_limit, &susint_funcs);

   menu_Flush(menu);
```

```
   sed = sed_Open(menu);
   sed_SetColors(sed, menu_color, menu_color, field_color);
   sed_SetPosition(sed, 5, 15);

   sed_Push(sed);
   sed_Repaint(sed);
   show_key_help(&key_menu, &key_sed, config_help_string);

/* ================================================================== */
/*                    This is the Main Control Loop                   */
/*                  This loop terminates on ESC or F10.               */
/* ================================================================== */

   do
      {
      do
         {
/* ================================================================== */
/*                        Handle the field entry.                     */
/* ================================================================== */

         ret = sed_Go(sed);

/* ================================================================== */
/*  Help Requested - Call centralized help function                   */
/* ================================================================== */

         if (ret == FN1)
            {
            compile_help(CONFIGURE_SYSTEM);
            sed_SetBaton(sed, -1);
            }
         }
      while ((ret) && (ret != FN10));
      }
   while (verify_ypc(scratch_limit));


/* ================================================================== */
/*                           End Main Loop                            */
/* ================================================================== */

/* ================================================================== */
/*  If the FN10 key was pressed, the user accepted the current status of */
/*  the configuration variables.  Copy the scratch variables back to the */
/*  original ones and write them out to the configuration file.       */
/* ================================================================== */

   if (ret == FN10)
      {

/* ================================================================== */
/*  Cscape provides a macro interface to their ticker string that returns */
/*  the pointer to the string part of the ticker string.  This is padded */
/*  with spaces and must be truncated.                                */
/* ================================================================== */
      strcpy(config.working_dir, tick_GetString(scratch_dir));

      len = strlen(config.working_dir);

      for (i = 0 ; i < len ; i++)
         if (config.working_dir[i] == ' ')
            {
            config.working_dir[i] = '\0';
            break;
```



```
            }

      config.print_device = device_choice.number;
      config.default_mode = defaults_choice.number;

      if (! verify_ypc(scratch_limit))
         config.years_passed_closure = scratch_limit;

      len = strlen(config.working_dir);

/* ================================================================== */
/*  The convention chosen is to make sure all directory paths do not end */
/*  in a backslash.  In this function, the backslash is put on when   */
/*  first entering and taken off here.  (Also, this code is repeated  */
/*  this if is finish but there is no backslash for it to find so this */
/*  is OK.  I need it formatted correctly when I write it to disk.    */
/* ================================================================== */

      if (*(config.working_dir + len - 1) == '\\')
         *(config.working_dir + len - 1) = '\0';

/* ================================================================== */
/*                      One of the misc_utl functions.                */
/* ================================================================== */
      capital_string(config.working_dir);

/* ================================================================== */
/*  Piece the filename together and write the configuration information */
/*  to LANDFILL.CFG.                                                  */
/* ================================================================== */

      strcpy(scratch_dir, config.landfill_dir);
      strcat(scratch_dir, "LANDFILL.CFG");

      config_file = fopen(scratch_dir, "w");

      if (config_file != NULL)
         {
         fprintf(config_file, "%s\n", config.working_dir);
         fprintf(config_file, "%d\n", config.print_device);
         fprintf(config_file, "%d\n", config.default_mode);
         fprintf(config_file, "%d\n", config.years_passed_closure);

         fclose(config_file);
         }
      }

   len = strlen(config.working_dir);

   if (*(config.working_dir + len - 1) == '\\')
      *(config.working_dir + len - 1) = '\0';

/* ================================================================== */
/*                  All done!  Delete the menu and return.            */
/* ================================================================== */

   destroy_key_help(key_menu, key_sed);
   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);
   return(ret);
}
```

```
int verify_ypc(int years)
[
   if (years > MAX_YEARS_PAST_CLOSURE)
      {
      verify_error(12);
      return(1);
      }

   return(0);
]
```

```
/* ========================================================================== */
/*                                  CREDITS.C                                 */
/* ========================================================================== */
/*  This file contains the code for displaying the title screen for the       */
/*  landfill program.                                                         */
/* ========================================================================== */
/*                            Created under Contract                          */
/*                                    to                                      */
/*                              EPA/AEERL/GECD/EMB                            */
/*      -                                                                     */
/*                                    by                                      */
/*                                                                            */
/*                              Radian Corporation                            */
/*                           Research Triangle Park, NC                       */
/* ========================================================================== */
/* ========================================================================== */
/*                              Revision History                              */
/* ========================================================================== */
/*    07/03/90    rlb       Initial Implementation                            */
/* ========================================================================== */

#include <stdio.h>
#include <cscape.h>
#include <string.h>
#include "landfill.h"
#include "helpasgn.h"
#include "colors.h"

/* ========================================================================== */
/*         This function defines and displays the title screen.               */
/* ========================================================================== */

void credits()
{
menu_type   menu, key_menu;
sed_type    sed, key_sed;
int old_cursor;
char version_string[35];

   old_cursor = vid_GetCursorType();
   vid_SetCursorType(CURSOR_NONE);

   strcenter(strcpy(version_string, TITLE_STRING), 33);

   menu = menu_Open();

   menu_Printf(menu, "@p[0,0]@c[%c]+@[33,-]+", menu_color);
   menu_Printf(menu, "@p[1,0]|");
   menu_Printf(menu, "@p[1,34]|");
   menu_Printf(menu, "@p[2,0]|");
   menu_Printf(menu, "@p[2,13]Landfills");
   menu_Printf(menu, "@p[2,34]|");
   menu_Printf(menu, "@p[3,0]|");
   menu_Printf(menu, "@p[3,11]Air Emissions");
   menu_Printf(menu, "@p[3,34]|");
   menu_Printf(menu, "@p[4,0]|");
   menu_Printf(menu, "@p[4,10]Estimation Model");
   menu_Printf(menu, "@p[4,34]|");
   menu_Printf(menu, "@p[5,0]|");
   menu_Printf(menu, "@p[5,34]|");
   menu_Printf(menu, "@p[6,0]|");
   menu_Printf(menu, "@p[6,34]|");
   menu_Printf(menu, "@p[7,0]|@fp[@[33,#]]|", version_string, &string_funcs);
   menu_Printf(menu, "@p[8,0]|");
   menu_Printf(menu, "@p[8,34]|");
   menu_Printf(menu, "@p[9,0]|");
```



```
   menu_Printf(menu, "@p[9,34]|");
   menu_Printf(menu, "@p[10,0]+@[33,-]+");

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, menu_color, menu_color, menu_color);
   sed_SetPosition(sed, 7, 22);

   sed_Push(sed);
   sed_Repaint(sed);
   show_key_help(&key_menu, &key_sed, any_key_string);

   kb_Read();

   destroy_key_help(key_menu, key_sed);
   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);

   vid_SetCursorType(old_cursor);

   return;
}
```

```
/* ========================================================================== */
/*                                  DISPRSLT.C                                */
/* ========================================================================== */
/*  This file contains the code for displaying the output method              */
/*  selection menu.                                                           */
/* ========================================================================== */
/*                              Created under Contract                        */
/*                                       to                                   */
/*                                 EPA/AEERL/GECD/EMB                         */
/*                                                                            */
/*                                       by                                   */
/*                                                                            */
/*                                 Radian Corporation                         */
/*                              Research Triangle Park, NC                    */
/* ========================================================================== */
/* ========================================================================== */
/*                                Revision History                            */
/* ========================================================================== */
/*    07/03/90   rlb        Initial Implementation                            */
/* ========================================================================== */

/* ========================================================================== */
/*  This function defines and presents the menu to allow the user to          */
/*  select the report method : tabular or on-screen graphic.                  */
/* ========================================================================== */

#include <stdio.h>
#include <cscape.h>
#include <scancode.h>
#include "helpasgn.h"
#include "colors.h"

int display_results_menu(choice)
int choice;
{
menu_type   menu, key_menu;
sed_type    sed, key_sed;
int         ret;

   menu = menu_Open();

   buildbox(menu, menu_color, 10, 29);
   menu_Printf(menu, "@p[1,4]Display Results Menu");
   menu_Printf(menu, "@p[4,1]@f[       Tabular Report        ]",
     NULL, &menu_funcs);
   menu_Printf(menu, "@p[5,1]@f[     On-screen Graphic       ]",
     NULL, &menu_funcs);
   menu_Printf(menu, "@p[7,1]@f[       Previous Menu         ]",
     NULL, &menu_funcs);

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, menu_color, menu_color, field_color);
   sed_SetPosition(sed, 7, 25);

   sed_Push(sed);

   sed_GotoField(sed, choice);

   sed_Repaint(sed);
   show_key_help(&key_menu, &key_sed, generic_menu_help_string);

/* ========================================================================== */
/*                                   Main Loop                                */
/*  This loop continues until the ret code from sed_Go is less than the       */
```



```
/*  field count.  sed_Go returns the menu field number of the item that       */
/*  generated the exit, the value of a special exit key, or 0 if ESC was      */
/*  pressed.                                                                  */
/* ========================================================================== */

   do
      {
/* ========================================================================== */
/*                              Get the field input                           */
/* ========================================================================== */
      ret = sed_Go(sed);

/* ========================================================================== */
/*         Call the centralize help facility if the FN1 was pressed.          */
/* ========================================================================== */

      if (ret == FN1)
         {
         compile_help(DISPLAY_RESULTS_MENU);
         sed_SetBaton(sed, -1);
         }
      }
   while (ret > sed_GetFieldCount(sed));

   destroy_key_help(key_menu, key_sed);
   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);
   return(ret);
}
```

```
/* ========================================================================== */
/*                                  DO_GRAPH.C                                */
/* ========================================================================== */
/*  This file contains the code for displaying the results graphically        */
/* ========================================================================== */
/*                             Created under Contract                         */
/*                                       to                                   */
/*                               EPA/AEERL/GECD/EMB                           */
/*                                                                            */
/*                                       by                                   */
/*                                                                            */
/*                               Radian Corporation                           */
/*                            Research Triangle Park, NC                      */
/* ========================================================================== */
/* ========================================================================== */
/*                               Revision History                             */
/* ========================================================================== */
/*    07/03/90    rlb        Initial Implementation                           */
/* ========================================================================== */

#include <stdio.h>
#include <cscape.h>
#include <malloc.h>
#include <math.h>
#include <ctype.h>
#include <graph.h>
#include <pgchart.h>
#include <graph.h>
#include "landfill.h"
#include "helpasgn.h"

#define WORK_SPACE 15

char *legend1 = "Estimated Emissions";
char *legend2 = "Projected Emissions";

/* ========================================================================== */
/*              This function actually displays the graphic                   */
/* ========================================================================== */

int display_onscreen_graph(x, y, current, count, title, units)
int *x, count, current;
double *y;
char *title, *units;
{
double max_y, factor, map_x, map_y;
struct videoconfig screen;
chartenv env, titles;
int i, j = 0, power, code, current_index;
int lx1, lx2, ly1, ly2;

float *y_copy, *x_copy, range, base;

/* ========================================================================== */
/*     Create local copies of the data compatible with the presentation       */
/*                          graphics functions                                */
/* ========================================================================== */

   x_copy = (float *) calloc((size_t) count, sizeof(float));
   y_copy = (float *) calloc((size_t) count, sizeof(float));

   if ((x_copy == NULL) || (y_copy == NULL))
      show_mem_error("display_onscreen_graph", "xy_data");

/* ========================================================================== */
/*  Determine the maximum y value.  This will be used to autoscale the y      */
```



```
/*  axis.                                                                     */
/* ========================================================================== */

   for (max_y = y[0], i = 1 ; i < count ; i++)
      if (max_y < y[i]) max_y = y[i];

   if (max_y > 0.0000)
      {
      power = floor(log10(max_y)) - 1;
      factor = pow(10.0, (double) power);
      }
   else
      {
      factor = 1;
      power = 0;
      }

   current_index = current - x[0];

   for (i = 0 ; i < count ; i++)
      {
      x_copy[i] = (float) x[i];
      y_copy[i] = (float) y[i] / factor;
      }

/* ========================================================================== */
/*  Set the video mode to graphics.  If the video card does not allow         */
/*  an acceptable mode, return to the calling function.                       */
/* ========================================================================== */

   if (! set_video_mode(&screen)) return(1);

/* ========================================================================== */
/*                        Set up presentation graphics                        */
/* ========================================================================== */

   _pg_initchart();
   _pg_defaultchart(&env, _PG_SCATTERCHART, _PG_POINTANDLINE);

/* ========================================================================== */
/*   Turn off autoscale after analyzing so I can take care of some of         */
/*   the fundamental things myself.                                           */
/* ========================================================================== */

   env.yaxis.autoscale = env.xaxis.autoscale = FALSE;

/* ========================================================================== */
/*                Set up some of the basic scale definitions                  */
/* ========================================================================== */

   env.xaxis.grid = env.yaxis.grid = FALSE;
   env.xaxis.labeled = env.yaxis.labeled = TRUE;
   env.xaxis.rangetype = env.yaxis.rangetype = _PG_LINEARAXIS;
   env.xaxis.ticformat = env.yaxis.ticformat = _PG_DECFORMAT;

/* ========================================================================== */
/*   Determine the x scale extent, set the tick marks, and set the axis       */
/*   scale factor.                                                            */
/* ========================================================================== */

/*   env.xaxis.scalemax = (float)((((int) x_copy[count - 1] / 10) * 10) + 10); *
/

   range = x_copy[count - 1] - x_copy[0];
   base = range / pow(10.0, floor(log10(range)));
```

```
   if (base < 1.5) base = 1.0;
   if ((base >= 1.5) && (base < 3.0)) base = 2.0;
   if (base >= 3.0) base = 5.0;

   env.xaxis.ticinterval = base * pow(10.0, floor(log10(range)) - 1);

   env.xaxis.scalemin = floor((int) x_copy[0] / base) * base;
        env.xaxis.scalemax = ceil((double) x_copy[count - 1] / base) * base;

   env.xaxis.scalefactor = 1.0;
   env.xaxis.scaletitle.title[0] = '\0';

/* ================================================================== */
/*   Determine the y scale extent, set the tick marks, and set the axis  */
/*   scale factor.                                                      */
/* ================================================================== */

   env.yaxis.scalemin = 0.0;
   env.yaxis.scalemax = (float)((int)(max_y / factor) + 1);

   range = (env.yaxis.scalemax - env.yaxis.scalemin);

   base = range / pow(10.0, floor(log10(range)));

   if (base < 1.5) base = 1.0;
   if ((base >= 1.5) && (base < 3.5)) base = 2.0;
   if (base >= 3.5) base = 5.0;

   env.yaxis.ticinterval = base;

   env.yaxis.scalemax = ceil(env.yaxis.scalemax / base) * base;

   env.yaxis.scalefactor = 1;

   strcpy(env.xaxis.axistitle.title, "Year (* = Current Year)");

   sprintf(env.yaxis.axistitle.title, "Emission Rate");
        sprintf(env.yaxis.scaletitle.title, "1E%-2d %s", power, units);

   env.xaxis.axistitle.justify = env.yaxis.axistitle.justify =
      _PG_CENTER;

   strcpy(env.maintitle.title, "Projected ");
   strcat(env.maintitle.title, title);
   for (i = strlen(env.maintitle.title) - 1 ; i >= 0 ; i--)
      if (isalnum(env.maintitle.title[i]))
         {
         env.maintitle.title[i + 1] = '\0';
         break;
         }

   env.xaxis.ticdecimals = 0;
   env.yaxis.ticdecimals = 1;

   strcat(env.maintitle.title, " Emission Rate");

/* ================================================================== */
/*  First, analyze the chart.  This allows presentation graphics to set  */
/*  the chart parameters for me.  I reduce the chart space a little to  */
/*  allow a legend later.                                              */
/* ================================================================== */

        code = _pg_analyzescatter(&env, (float far *) (x_copy + current_index),
                                  (float far *) (y_copy + current_index), 1);

        env.chartwindow.border = 0;
```



```
        env.datawindow.border = 0;

        env.chartwindow.y2 -= WORK_SPACE;

/* ================================================================== */
/*  Now, draw the chart.  The only point I want explicitly place on the  */
/*  graph is the current year (labeled with a *).  I will take care of  */
/*  plotting the rest of the data.                                     */
/* ================================================================== */

   code = _pg_chartscatter(&env, (float far *) (x_copy + current_index),
                                 (float far *) (y_copy + current_index), 1);

/* ================================================================== */
/*  Use the information returned in the chartenv structure to determine  */
/*  the scale.                                                         */
/* ================================================================== */

   map_x = (double) (env.datawindow.x2 - env.datawindow.x1)
                    / (double) (env.xaxis.scalemax - env.xaxis.scalemin);
   map_y = (double) (env.datawindow.y1 - env.datawindow.y2)
                    / (double) (env.yaxis.scalemax - env.yaxis.scalemin);

/* ================================================================== */
/*       Plot the estimated emissions line in color 1 and a solid line  */
/* ================================================================== */

   _setlinestyle(0xFFFF);

        _setcolor(1);

        _moveto((short) (env.datawindow.x1 +
                ((x_copy[0] - env.xaxis.scalemin) * map_x)),
                   (short)(env.datawindow.y2 + (map_y * y_copy[0])));

   for (i = 1 ; i <= current_index ; i++)
      _lineto((short) (env.datawindow.x1 +
                (int) (map_x * (x_copy[i] - env.xaxis.scalemin))),
                   (short) (env.datawindow.y2 + (int) (map_y * y_copy[i])));

/* ================================================================== */
/*       Plot the projected emissions line in color 2 and a dotted line  */
/* ================================================================== */

        _setcolor(2);

   _setlinestyle(0x5555);

   for (i = current_index + 1 ; i < count; i++)
      _lineto((short) (env.datawindow.x1 +
                (int) (map_x * (x_copy[i] - env.xaxis.scalemin))),
                   (short) (env.datawindow.y2 + (int) (map_y * y_copy[i])));

/* ================================================================== */
/*  Put a legend on the chart.  In order to use the graphics font used by  */
/*  presentation graphics and prevent clipping, this requires creating a  */
/*  chart with presentation graphics.                                  */
/* ================================================================== */

   _pg_defaultchart(&titles, _PG_SCATTERCHART, _PG_POINTANDLINE);

        lx1 = titles.chartwindow.x1 = env.chartwindow.x1;
        lx2 = titles.chartwindow.x2 = env.chartwindow.x2;
        ly1 = titles.chartwindow.y1 = env.chartwindow.y2;
        ly2 = titles.chartwindow.y2 = env.chartwindow.y2 + WORK_SPACE;
```

```
        _pg_hlabelchart(&titles, lx2 / 2 - 8 * (2 + strlen(legend1)),
                  5, 1, legend1);

        _pg_hlabelchart(&titles, lx2 / 2 + 48, 5, 1, legend2);

    _setlinestyle(0xFFFF);
        _setcolor(1);

    _moveto((short)(lx2 / 2 - 8 * (4 + strlen(legend1))), ly1 + 9);
    _lineto((short)(lx2 / 2 - 8 * (8 + strlen(legend1))), ly1 + 9);

    _setlinestyle(0x5555);
        _setcolor(2);

    _moveto((short)(lx2 / 2 + 32), ly1 + 9);
    _lineto((short)(lx2 / 2), ly1 + 9);

/* ========================================================================= */
/*          If plotting was successful, wait for a key and clean up          */
/* ========================================================================= */

    if (code != 0)
       {
       printf("Chartscatter returned %d\n", code);
       getchar();
       }
    else
       kb_Read();

    _setvideomode(_DEFAULTMODE);

    free(x_copy);
    free(y_copy);

    return(0);
}



/* ========================================================================= */
/*  This function puts the card in graphics mode.  Supported cards are       */
/*  VGA, EGA, CGA, and Hercules (with MSHERC installed).                     */
/* ========================================================================= */

int set_video_mode(screen)
struct videoconfig *screen;
{
int ret = 0;

    _getvideoconfig(screen);

    switch (screen->adapter)
       {
                case _OCGA:
                case _OEGA:
                case _OVGA:
          ret = _setvideomode(_ORESCOLOR);
          break;

       case _CGA:
                ret = _setvideomode(_HRESBW);

          break;

       case _EGA:
              ret = _setvideomode(_ERESCOLOR);
```



```
          break;

       case _SVGA:
       case _VGA:
       case _MCGA:
          ret = _setvideomode(_VRES16COLOR);
          break;

       case _HGC:
          ret = _setvideomode(_HERCMONO);
          break;

       default:
          _setvideomode(_DEFAULTMODE);
       }

    if (! ret) no_card_support();
    return(ret);
}


/* ========================================================================= */
/*       This function pops up an error if the card is not supported.       */
/* ========================================================================= */

int no_card_support()
{
menu_type   menu, key_menu;
sed_type    sed, key_sed;
int         ret;

    menu = menu_Open();

    menu_Printf(menu, "@p[0,0]@c[\x07]+@[31,-]+");
    menu_Printf(menu, "@p[1,0]|     Monitor Not Supported     |");
    menu_Printf(menu, "@p[2,0]|@[31,-]|");
    menu_Printf(menu, "@p[3,0]|");
    menu_Printf(menu, "@p[3,32]|");
    menu_Printf(menu, "@p[4,0]|   Your monitor card is not    |");
    menu_Printf(menu, "@p[5,0]|   supported by this system.   |");
    menu_Printf(menu, "@p[6,0]|");
    menu_Printf(menu, "@p[6,32]|");
    menu_Printf(menu, "@p[7,0]+@[31,-]+");

    menu_Flush(menu);

    sed = sed_Open(menu);
    sed_SetColors(sed, '\x07', '\x07', '\x70');
    sed_SetPosition(sed, 9, 23);

    sed_Push(sed);
    sed_Repaint(sed);
    show_key_help(&key_menu, &key_sed, any_key_string);

    kb_Read();

    destroy_key_help(key_menu, key_sed);
    sed_Pop(sed);

    menu_Close(menu);
    sed_Close(sed);
    return(ret);
}
```

```
/* ========================================================================== */
/*                                 EDITMAIN.C                                 */
/* ========================================================================== */
/*  This file the major control structures for the edit portion of the        */
/*  Landfill Air Emmisions model.                                             */

/* ========================================================================== */
/*                           Created under Contract                           */
/*                                    to                                      */
/*                             EPA/AEERL/GECD/EMB                             */
/*                                                                            */
/*                                    by                                      */
/*                                                                            */
/*                             Radian Corporation                             */
/*                          Research Triangle Park, NC                        */
/* ========================================================================== */
/* ========================================================================== */
/*                             Revision History                               */
/* ========================================================================== */
/*    07/03/90   rlb       Initial Implementation                             */
/*    01/21/94   rlb       Changed include_air_toxics flag comparison to      */
/*                           NO_AT to support landfill disposal type.         */
/* ========================================================================== */

#include <stdio.h>
#include <cscape.h>
#include <scancode.h>
#include "landfill.h"
#include "defaults.h"

boolean edit_main(landfill_info, model_info, k_calc_info, calc_possible)
struct landfill_parms *landfill_info;
struct model_parms *model_info;
struct k_calc_parms *k_calc_info;
boolean *calc_possible;
{
int choice = 1, k_choice = 1, ret;
boolean changed = FALSE;

extern int edit_menu(int);
extern int air_toxics_menu(struct air_toxic_structure **, int *);
extern int edit_chem_comp(struct model_parms *);
extern int edit_periods(struct landfill_parms *);
extern int l0_selection_menu(double *);
extern int k_selection_menu(int);
extern int edit_k_value(double *);
extern int warn_about_2e();
extern int calculate_k_menu(struct k_calc_parms *, struct model_parms *);

/* ========================================================================== */
/* Essentially, this function is one big loop which activates all of the     */
/* menus coordinates the interrelations of the menus.                         */
/* ========================================================================== */

   while (choice = edit_menu(choice - 1))
      {
      switch (choice)
         {
         case 0:
            choice = 1;
            break;

         case 1:
            if (ret = edit_chem_comp(model_info))
               {
               if (model_info->include_air_toxics != NO_AT)
```



```
                  air_toxics_menu(&(model_info->air_toxics),
                                  &(model_info->number_air_toxics));
               else
                  {
                  if (model_info->air_toxics != NULL)
                     free(model_info->air_toxics);

                  model_info->air_toxics = NULL;

                  model_info->number_air_toxics = 0;
                  }
               }

            break;

         case 2:
            switch (ret = k_choice = k_selection_menu(k_choice - 1))
               {
               case 1:
                  if (config.default_mode == AP_MODE)
                     model_info->k = DEFAULT_AP42_K;
                  else model_info->k = DEFAULT_REGULATORY_K;
                  break;

               case 2:
                  ret = edit_k_value(&(model_info->k));

                  break;

               case 3:
                  if (warn_about_2e() != ESC)
                     ret = calculate_k_menu(k_calc_info, model_info);

                  break;

               default:
                  k_choice = 1;
               }
            break;

         case 3:
            ret = l0_selection_menu(&(model_info->l0));

            break;

         case 4:
            ret = edit_periods(landfill_info);
            if (ret) *calc_possible = TRUE;
            break;

         case 5:
            return(changed);
         }

      if (ret) changed = TRUE;
      }

   return(changed);
}
```

```
/* ========================================================================== */
/*                                EDITMENU.C                                  */
/* ========================================================================== */
/*      This file contains the code for displaying edit menu                  */
/* ========================================================================== */
/*                           Created under Contract                           */
/*                                   to                                       */
/*                             EPA/AEERL/GECD/EMB                             */
/*                                                                            */
/*                                   by                                       */
/*                                                                            */
/*                             Radian Corporation                             */
/*                          Research Triangle Park, NC                        */
/* ========================================================================== */
/* ========================================================================== */
/*                           Revision History                                 */
/* ========================================================================== */
/*    07/03/90   rlb       Initial Implementation                             */
/* ========================================================================== */

#include <stdio.h>
#include <cscape.h>
#include <scancode.h>
#include "helpasgn.h"
#include "colors.h"

int edit_menu(choice)
int choice;
{
menu_type   menu, key_menu;
sed_type    sed, key_sed;
int         ret;

void show_edit_help();

   menu = menu_Open();

   buildbox(menu, menu_color, 12, 36);
   menu_Printf(menu, "@p[1,10]Edit Study Data");

   menu_Printf(menu, "@p[4,1]@f[@[7, ]Chemical Composition@[7, ]]",
     NULL, &menu_funcs);
   menu_Printf(menu, "@p[5,1]@f[    Methane Rate Constant  (k)    ]",
     NULL, &menu_funcs);
   menu_Printf(menu, "@p[6,1]@f[      Methane Potential (Lo)      ]",
     NULL, &menu_funcs);
   menu_Printf(menu, "@p[7,1]@f[@[9, ]Operational Data@[9, ]]",
     NULL, &menu_funcs);
   menu_Printf(menu, "@p[9,1]@f[@[10, ]Previous Menu@[11, ]]",
     NULL, &menu_funcs);

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, menu_color, menu_color, field_color);
   sed_SetPosition(sed, 6, 22);

   sed_Push(sed);

   sed_GotoField(sed, choice);

   sed_Repaint(sed);
   show_key_help(&key_menu, &key_sed, generic_menu_help_string);

   do
      {
```



```
      ret = sed_Go(sed);

      if (ret == FN1)
         {
         compile_help(EDIT_MENU);
         sed_SetBaton(sed, -1);
         }
      }
   while (ret > sed_GetFieldCount(sed));

   destroy_key_help(key_menu, key_sed);
   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);
   return(ret);
}
```

```
/* ========================================================================== */
/*                                   EXIT.C                                   */
/* ========================================================================== */
/*  This file contains the code for displaying the exit selection menu        */
/* ========================================================================== */
/*                            Created under Contract                          */
/*                                      to                                    */
/*                              EPA/AEERL/GECD/EMB                            */
/*                                                                            */
/*                                      by                                    */
/*                                                                            */
/*                              Radian Corporation                            */
/*                           Research Triangle Park, NC                       */
/* ========================================================================== */
/* ========================================================================== */
/*                              Revision History                              */
/* ========================================================================== */
/*    07/03/90   rlb       Initial Implementation                             */
/* ========================================================================== */

#include <stdio.h>
#include <cscape.h>
#include <scancode.h>
#include "colors.h"


int exit_program()
[
menu_type   menu;
sed_type    sed;
int         ret;

   menu = menu_Open();

   buildbox(menu, menu_color, 8, 28);

   menu_Printf(menu, "@p[1,4]Do you wish to exit?");
   menu_Printf(menu, "@p[4,1]@f[@[11, ]Yes@[12, ]]",
     NULL, &menu_funcs);
   menu_Printf(menu, "@p[5,1]@f[@[11, ]No@[13, ]]",
     NULL, &menu_funcs);

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, menu_color, menu_color, field_color);
   sed_SetPosition(sed, 8, 25);

   sed_Push(sed);
   sed_Repaint(sed);

   do ret = sed_Go(sed);
   while (ret > sed_GetFieldCount(sed));

   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);
   return(ret);
]
```

```
/* ========================================================================= */
/*                                FINDFLS.C                                  */
/* ========================================================================= */
/*  This file contains the code for obtaining the list of study files in     */
/*  the current working directory.                                           */
/* ========================================================================= */
/*                           Created under Contract                          */
/*                                    to                                     */
/*                             EPA/AEERL/GECD/EMB                            */
/*                                                                           */
/*                                    by                                     */
/*                                                                           */
/*                             Radian Corporation                            */
/*                          Research Triangle Park, NC                       */
/* ========================================================================= */
/* ========================================================================= */
/*                             Revision History                              */
/* ========================================================================= */
/*    07/03/90   rlb       Initial Implementation                            */
/* ========================================================================= */

#include <stdio.h>
#include <dos.h>
#include <malloc.h>
#include "landfill.h"

unsigned compile_file_list(buffer)
char (**buffer)[13];
{
extern struct system_parms config;
struct find_t file_info;
unsigned count = 0, index = 0;
char path[256];

/* ========================================================================= */
/*                  Piece together the path + filename.                      */
/* ========================================================================= */

   strcpy(path, config.working_dir);
   strcat(path, "\\");

   strcat(path, "*.PRM");

/* ========================================================================= */
/*   Get a count of the number of files present in the working directory     */
/*   which match the file specification.  This will be used to malloc        */
/*   space for the list.                                                     */
/* ========================================================================= */

   if (! _dos_findfirst(path, _A_NORMAL, &file_info)) count++;
   while (! _dos_findnext(&file_info)) count++;

/* ========================================================================= */
/*    If there are no files, no need to go any further.  Return a NULL       */
/*    pointer and a zero count.                                              */
/* ========================================================================= */

   if (! count)
      {
      *buffer = NULL;
      return(0);
      }

/* ========================================================================= */
/*    Having gotten the file count, make space available for the list.       */
/* ========================================================================= */
```



```
   *buffer = (char (*)[13]) calloc((size_t) count, 13 * sizeof(char));

   if (*buffer == NULL)
      show_mem_error("compile_file_list", "buffer");

/* ========================================================================= */
/*                Now that we have space, fill the memory.                   */
/* ========================================================================= */

   if (! _dos_findfirst(path, _A_NORMAL, &file_info))
      strcpy((*buffer)[index++], file_info.name);

   while (! _dos_findnext(&file_info))
      strcpy((*buffer)[index++], file_info.name);

/* ========================================================================= */
/*          Now, the list has been compiled.  Return the count.              */
/* ========================================================================= */

   return(count);
}
```

```
/* ========================================================================= */
/*                                FLSELECT.C                                 */
/* ========================================================================= */
/*  This file contains the code for displaying the file selector menu.       */
/* ========================================================================= */
/*                           Created under Contract                          */
/*                                    to                                     */
/*                             EPA/AEERL/GECD/EMB                            */
/*                                                                           */
/*                                    by                                     */
/*                                                                           */
/*                             Radian Corporation                            */
/*                          Research Triangle Park, NC                       */
/* ========================================================================= */
/* ========================================================================= */
/*                            Revision History                               */
/* ========================================================================= */
/*    07/03/90   rlb       Initial Implementation                            */
/* ========================================================================= */

#include <stdio.h>
#include <cscape.h>
#include <scancode.h>
#include <string.h>
#include "landfill.h"
#include "helpasgn.h"
#include "colors.h"

#define UP_ARROW      24
#define DOWN_ARROW    25
#define SPACE         32

int file_selector(list, count)
char (*list)[13];
unsigned count;
{
menu_type   menu, list_menu, key_menu;
sed_type    sed, list_sed, key_sed;
int         i, ret, right_pad, curr_offset = 0, old_offset, menu_height;
int down_indicator = SPACE, up_indicator = SPACE;
char format[80], *pointer;

   if (count > 7) down_indicator = DOWN_ARROW;

   menu = menu_Open();

   menu_Printf(menu, "@p[0,0]@c[%c]+@[24,-]+", menu_color);
   menu_Printf(menu, "@p[1,0]¦ Choose a Study to Edit ¦");
   menu_Printf(menu, "@p[2,0]¦--@[22,-]¦");
   menu_Printf(menu, "@p[3,0]¦ ¦");
   menu_Printf(menu, "@p[3,25]¦");
   menu_Printf(menu, "@p[4,0]¦ ¦");
   menu_Printf(menu, "@p[4,25]¦");
   menu_Printf(menu, "@p[5,0]¦ ¦");
   menu_Printf(menu, "@p[5,25]¦");
   menu_Printf(menu, "@p[6,0]¦ ¦");
   menu_Printf(menu, "@p[6,25]¦");
   menu_Printf(menu, "@p[7,0]¦ ¦");
   menu_Printf(menu, "@p[7,25]¦");
   menu_Printf(menu, "@p[8,0]¦ ¦");
   menu_Printf(menu, "@p[8,25]¦");
   menu_Printf(menu, "@p[9,0]¦ ¦");
   menu_Printf(menu, "@p[9,25]¦");
   menu_Printf(menu, "@p[10,0]¦ ¦");
   menu_Printf(menu, "@p[10,25]¦");
```



```
   menu_Printf(menu, "@p[11,0]+--@[22,-]+");

   menu_Printf(menu, "@p[3,1]@fp[#]", &down_indicator, &char_funcs);
   menu_Printf(menu, "@p[10,1]@fp[#]", &up_indicator, &char_funcs);

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, menu_color, menu_color, menu_color);
   sed_SetPosition(sed, 6, 26);

   list_menu = menu_Open();

   for (i = 0 ; i < 8 ; i++)
      menu_Printf(list_menu, "@p[%d,0]@c[%c]@[37, ]", i, menu_color);

   for (i = 0 ; i < count ; i++)
      {
      pointer = strstr(list[i], ".");
      *pointer = '\0';

      right_pad = 22 - strlen(list[i]);

      sprintf(format, "@p[%d,0]@c[%c]@f[ %s@[%d, ]]", i, menu_color,
                      list[i], (right_pad >= 0 ? right_pad : 0));

      menu_Printf(list_menu, format, NULL, &menu_funcs);
      }

   menu_Flush(list_menu);

   list_sed = sed_Open(list_menu);
   sed_SetColors(list_sed, menu_color, menu_color, field_color);
   sed_SetPosition(list_sed, 9, 29);

   sed_SetWidth(list_sed, 22);
   sed_SetHeight(list_sed, 8);

   menu_height = sed_GetMenuHeight(list_sed);

   sed_Push(sed);
   sed_Repaint(sed);
   show_key_help(&key_menu, &key_sed, generic_menu_help_string);

   sed_Repaint(list_sed);

   sed_SetActive(list_sed, TRUE);

   sed_GotoFirstField(list_sed);

   sed_DoFieldSenter(list_sed, sed_GetFieldNo(list_sed));
   sed_DoFieldFenter(list_sed, sed_GetFieldNo(list_sed));

   do
      {
      sed_DoFieldFkey(list_sed, sed_GetFieldNo(list_sed));

      ret = sed_GetBaton(list_sed);

      switch (ret)
         {
         case FN1:
            compile_help(FILE_SELECTOR);
            sed_SetBaton(list_sed, -1);
            break;
         }
```

```
      curr_offset = sed_GetYoffset(list_sed);

      if (curr_offset) up_indicator = UP_ARROW;
      else up_indicator = SPACE;

      if ((curr_offset + 8) < menu_height) down_indicator = DOWN_ARROW;
      else down_indicator = SPACE;

      if (curr_offset != old_offset)
         {
         sed_RepaintFields(sed);
         old_offset = curr_offset;
         }
      }
   while ((ret > sed_GetFieldCount(list_sed)) || (ret == -1));

   destroy_key_help(key_menu, key_sed);
   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);

   menu_Close(list_menu);
   sed_Close(list_sed);

   return(ret);
}
```

```
/* ========================================================================= */
/*                                  GETOUT.C                                 */
/* ========================================================================= */
/*  This file contains the code for displaying the output filename entry     */
/*  menu.                                                                    */
/* ========================================================================= */
/*                            Created under Contract                         */
/*                                      to                                   */
/*                              EPA/AEERL/GECD/EMB                           */
/*                                                                           */
/*                                      by                                   */
/*                                                                           */
/*                              Radian Corporation                           */
/*                           Research Triangle Park, NC                      */
/* ========================================================================= */
/* ========================================================================= */
/*                            Revision History                               */
/* ========================================================================= */
/*    07/03/90   rlb        Initial Implementation                           */
/* ========================================================================= */

#include <stdio.h>
#include <cscape.h>
#include <scancode.h>
#include "landuser.h"
#include "helpasgn.h"
#include "colors.h"

int get_output_filename(filename)
char *filename;
{
menu_type   menu, key_menu;
sed_type    sed, key_sed;
int         ret;

   reset_insert_status();

   menu = menu_Open();

   menu_Printf(menu, "@p[0,0]@c[%c]+@[35,-]+", menu_color);
   menu_Printf(menu, "@p[1,0]¦");
   menu_Printf(menu, "@p[1,8]Enter Output Filename");
   menu_Printf(menu, "@p[1,36]¦");
   menu_Printf(menu, "@p[2,0]¦@[35,-]¦");
   menu_Printf(menu, "@p[3,0]¦");
   menu_Printf(menu, "@p[3,36]¦");
   menu_Printf(menu, "@p[4,0]¦");
   menu_Printf(menu, "@p[4,36]¦");
   menu_Printf(menu, "@p[5,0]¦");
   menu_Printf(menu, "@p[5,36]¦");
   menu_Printf(menu, "@p[6,0]+@[35,-]+");
   menu_Printf(menu, "@p[4,13]@f[@[12,#]]",
     filename, &dosfile_funcs);

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, menu_color, menu_color, field_color);
   sed_SetPosition(sed, 9, 21);

   sed_Push(sed);
   sed_Repaint(sed);
   show_key_help(&key_menu, &key_sed, studname_help_string);

   do
      {
```



```
      ret = sed_Go(sed);

      if (ret == FN1)
         {
         compile_help(GET_OUTPUT_FILENAME);
         sed_SetBaton(sed, -1);
         }
      }
   while ((ret > sed_GetFieldCount(sed)) ||
          ((strlen(filename) == 0) && (ret)));

   destroy_key_help(key_menu, key_sed);
   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);
   return(ret);
}
```

```
/* ========================================================================== */
/*                                  HELP_SCR.C                                */
/* ========================================================================== */
/*  This file contains the code for displaying help information for a         */
/*  specific menu.                                                            */
/* ========================================================================== */
/*                            Created under Contract                          */
/*                                      to                                    */
/*                                EPA/AEERL/GECD/EMB                          */
/*                                                                            */
/*                                      by                                    */
/*                                                                            */
/*                                 Radian Corporation                         */
/*                              Research Triangle Park, NC                    */
/* ========================================================================== */
/* ========================================================================== */
/*                                Revision History                            */
/* ========================================================================== */
/*    07/03/90   rlb       Initial Implementation                             */
/* ========================================================================== */

#include <stdio.h>
#include <cscape.h>
#include <scancode.h>
#include <string.h>
#include <time.h>
#include "landfill.h"
#include "landuser.h"
#include "helpasgn.h"
#include "colors.h"

#define DELAY 2 /* Delay time for help warning messages in seconds */

/* ========================================================================== */
/*      IBM specific values for character codes for various symbols.          */
/* ========================================================================== */

#define UP_ARROW      24
#define DOWN_ARROW    25
#define SPACE         32

/* ========================================================================== */
/*  This function positions the file pointer at the appropriate location     */
/* in help file and passes control to the function that actually displays     */
/* the information.                                                           */
/* ========================================================================== */

int compile_help(sed_id)
int sed_id;
{
char buffer[256], filename[256];
int current_sed;
char *help_lines, *help_level;
FILE *help_file;
extern struct system_parms config;

void display_no_help(), display_missing_help();

/* ========================================================================== */
/*  The help file is expected to be in the directory containing the           */
/*  program.  If it is not found, display the help not found popup.           */
/* ========================================================================== */

   strcpy(filename, config.landfill_dir);
   strcat(filename, "LANDFILL.HLP");
```



```
   help_file = fopen(filename, "r");

   if (help_file == NULL)
      {
      display_no_help();

      return(0);
      }

/* ========================================================================== */
/*  Start from the beginning and search until the symbol '@@' is found.       */
/*  This signals the beginning of help information and the number that        */
/*  follows indicates which help entry it is.  Look through the file         */
/*  until the correct entry is found.  If it is not, display the missing      */
/*  help popup.                                                               */
/* ========================================================================== */

   while (fgets(buffer, 255, help_file) != NULL)
      {
      help_level = strstr(buffer, "@@");
      if (help_level != NULL)
         {
         sscanf(help_level + 2, "%d", &current_sed);
         if (current_sed == sed_id) break;
         }
      }

   if (current_sed != sed_id)
      {
      display_missing_help(sed_id);

      return(1);
      }

/* ========================================================================== */
/*      Ok, the file is at the right place.  Build the help screen.           */
/* ========================================================================== */

   show_help_screen(help_file);

   return(0);
}




int show_help_screen(help_file)
FILE *help_file;
{
unsigned old_cursor;
menu_type   menu, view_menu, key_menu;
sed_type    sed, view_sed, key_sed;
int         ret, line = 0, left_space, right_space;
char up_indicator = SPACE, down_indicator = SPACE, string[255];
int curr_offset = 0 , old_offset = 0;

/* ========================================================================== */
/*                             Build the view box                             */
/* ========================================================================== */

   old_cursor = vid_GetCursorType();
   vid_SetCursorType(CURSOR_NONE);

   menu = menu_Open();

   build_scrollbox(menu, info_color, 24, 80);
```

```
   menu_Printf(menu, "@p[3,1]@f[#]",
     &down_indicator, &char_funcs);
   menu_Printf(menu, "@p[22,1]@f[#]",
     &up_indicator, &char_funcs);


   if (fgets(string, 255, help_file) != NULL)
      {
      string[strlen(string) - 1] = '\0';
      left_space = (int) (78 - strlen(string)) / 2 + 1;
      right_space = 78 - left_space - strlen(string);
      menu_Printf(menu, "@p[1,1]@c[\x70]@[%d, ]%s@[%d, ]@c[\x07]",
         left_space, string, right_space);
      }
   else
      {
      menu_Close(menu);
      return(0);
      }

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, info_color, info_color, info_color);
   sed_SetPosition(sed, 0, 0);

   sed_Push(sed);
   sed_Repaint(sed);
   show_key_help(&key_menu, &key_sed, tabular_report_key_help);

/* ========================================================================= */
/*                   Build the information to be viewed.                     */
/* ========================================================================= */

   view_menu = menu_Open();

   while (fgets(string, 255, help_file) != NULL)
      {
/* ========================================================================= */
/*  Keep searching until the beginning of the next help block is found or    */
/*  the end of file is reached.                                              */
/* ========================================================================= */

      if (strstr(string, "@@") == NULL)
         menu_Printf(view_menu, "@c[%c]%s", info_color, string);
      else
         break;
      }

   menu_Flush(view_menu);

   view_sed = sed_Open(view_menu);
   sed_SetColors(view_sed, info_color, info_color, info_color);
   sed_SetPosition(view_sed, 3, 3);

   sed_SetWidth(view_sed, 76);
   sed_SetHeight(view_sed, 20);

   line = sed_GetMenuHeight(view_sed);

   if (line > 20)
      {
      down_indicator = DOWN_ARROW;
      sed_RepaintFields(sed);
      }
```



```
   sed_Repaint(view_sed);

   do
      {
      ret = kb_Read();

      switch (ret)
         {
         case PGUP:
            sed_ScrollUp(view_sed, 18);
            break;

         case PGDN:
            sed_ScrollDown(view_sed, 18);
            break;

         case UP:
            sed_ScrollUp(view_sed, 1);
            break;

         case DOWN:
            sed_ScrollDown(view_sed, 1);
            break;
         }

      curr_offset = sed_GetYoffset(view_sed);

      if (curr_offset) up_indicator = UP_ARROW;
      else up_indicator = SPACE;

      if ((line - curr_offset) > 20) down_indicator = DOWN_ARROW;
      else down_indicator = SPACE;

      if (curr_offset != old_offset)
         {
         sed_RepaintFields(sed);
         old_offset = curr_offset;
         }
      }
   while (ret != ESC);

   destroy_key_help(key_menu, key_sed);
   sed_Pop(sed);

   menu_Close(view_menu);
   menu_Close(menu);

   sed_Close(view_sed);
   sed_Close(sed);

   vid_SetCursorType(old_cursor);

   return(ret);
}



void display_no_help()
{
menu_type   menu;
sed_type    sed;
time_t   org_count, curr_count;

   menu = menu_Open();
```

```
   menu_Printf(menu, "@p[0,0]@c[\x07]+@[32,-]+");
   menu_Printf(menu, "@p[1,0]¦");
   menu_Printf(menu, "@p[1,33]¦");
   menu_Printf(menu, "@p[2,0]¦     Help File Not Present");
   menu_Printf(menu, "@p[2,33]¦");
   menu_Printf(menu, "@p[3,0]¦");
   menu_Printf(menu, "@p[3,33]¦");
   menu_Printf(menu, "@p[4,0]+@[32,-]+");

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, '\x07', '\x07', '\x70');
   sed_SetPosition(sed, 10, 23);

   sed_Push(sed);
   sed_Repaint(sed);

   time(&org_count);
   while (time(&curr_count) < org_count + DELAY);

   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);
   return;
}



void display_missing_help(number)
int number;
{
menu_type   menu;
sed_type    sed;
long        i;
time_t    org_count, curr_count;

   menu = menu_Open();

   menu_Printf(menu, "@p[0,0]+@[32,-]+");
   menu_Printf(menu, "@p[1,0]¦");
   menu_Printf(menu, "@p[1,33]¦");
   menu_Printf(menu, "@p[2,0]¦ No Help Available for Item     ¦");
   menu_Printf(menu, "@p[3,0]¦");
   menu_Printf(menu, "@p[3,33]¦");
   menu_Printf(menu, "@p[4,0]+@[32,-]+");
   menu_Printf(menu, "@p[2,29]@fp[###]", &number, &susint_funcs);

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, '\x07', '\x07', '\x70');
   sed_SetPosition(sed, 10, 23);

   sed_Push(sed);
   sed_Repaint(sed);

   time(&org_count);
   while (time(&curr_count) < org_count + DELAY);

   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);
   return;
```



```
}
```

```
/* ========================================================================== */
/*                                  KCALCMEN.C                                */
/* ========================================================================== */
/*  This file contains the code for displaying the entry screen of            */
/*  parameters necessary to calculate a k value.                              */
/* ========================================================================== */
/*                            Created under Contract                          */
/*                                    to                                      */
/*                              EPA/AEERL/GECD/EMB                            */
/*                                                                            */
/*                                    by                                      */
/*                                                                            */
/*                              Radian Corporation                            */
/*                          Research Triangle Park, NC                        */
/* ========================================================================== */
/* ========================================================================== */
/*                              Revision History                              */
/* ========================================================================== */
/*    07/03/90   rlb       Initial Implementation                             */
/* ========================================================================== */

#include <stdio.h>
#include <ctype.h>
#include <cscape.h>
#include <scancode.h>
#include "landuser.h"
#include "landfill.h"
#include "defaults.h"
#include "helpasgn.h"
#include "colors.h"

int calculate_k_menu(k_calc_info, model_info)
struct k_calc_parms *k_calc_info;
struct model_parms *model_info;
{
void (*old_fkey)();
boolean check_nonzero(struct k_calc_parms *, double);
menu_type   menu, key_menu;
sed_type    sed, key_sed;
int         ret;
double new_k;

void recalc_fkey();
double calculate_k(struct k_calc_parms *, struct model_parms *);

/* ========================================================================== */
/*  To implement auto recalc, I need to know when each value is change.       */
/*  Consequently, the normal function key function is swapped for a new       */
/*  one.                                                                      */
/* ========================================================================== */

   old_fkey = susdbl_funcs.fkey;
   susdbl_funcs.fkey = recalc_fkey;

   reset_insert_status();

   menu = menu_Open();

   buildbox(menu, menu_color, 21, 60);

   menu_Printf(menu, "@p[1,10]Calculate Methane Gas Generation Constant");
   menu_Printf(menu, "@p[4,2]Average Well Depth");
   menu_Printf(menu, "@p[4,50]ft");
   menu_Printf(menu, "@p[5,2]Average Stabilized");
   menu_Printf(menu, "@p[6,7]Radius of Influence");
   menu_Printf(menu, "@p[6,50]ft");
```



```
   menu_Printf(menu, "@p[7,2]Refuse Density");
   menu_Printf(menu, "@p[7,50]Mg/ft^3");
   menu_Printf(menu, "@p[8,2]Fraction of");
   menu_Printf(menu, "@p[9,7]Decomposable Refuse");
   menu_Printf(menu, "@p[10,2]Methane Generation");
   menu_Printf(menu, "@p[11,7]Potential (Lo)");
   menu_Printf(menu, "@p[11,50]ft^3/Mg");
   menu_Printf(menu, "@p[12,2]Average Stabilized");
   menu_Printf(menu, "@p[13,7]Flow Rate per Well");
   menu_Printf(menu, "@p[13,50]ft^3/min");
   menu_Printf(menu, "@p[14,2]Average Age of Refuse");
   menu_Printf(menu, "@p[14,50]yr");
   menu_Printf(menu, "@p[16,0]¦@[58,-]¦");
   menu_Printf(menu, "@p[18,5]Calculated Value of k");
   menu_Printf(menu, "@p[18,47]yr^-1");
   menu_Printf(menu, "@p[4,32]@f[@[17,#]]",
     &(k_calc_info->well_depth), &susdbl_funcs);
   menu_Printf(menu, "@p[6,32]@f[@[17,#]]",
     &(k_calc_info->radius), &susdbl_funcs);
   menu_Printf(menu, "@p[7,32]@f[@[17,#]]",
     &(k_calc_info->density), &susdbl_funcs);
   menu_Printf(menu, "@p[9,32]@f[@[17,#]]",
     &(k_calc_info->fraction), &susdbl_funcs);
   menu_Printf(menu, "@p[11,32]@f[@[17,#]]",
     &(model_info->l0), &susdbl_funcs);
   menu_Printf(menu, "@p[13,32]@f[@[17,#]]",
     &(k_calc_info->flow_rate), &susdbl_funcs);
   menu_Printf(menu, "@p[14,32]@f[@[17,#]]",
     &(k_calc_info->refuse_age), &susdbl_funcs);
   menu_Printf(menu, "@p[18,29]@fp[@[18,#]]",
     &new_k, &susdbl_funcs);

   new_k = model_info->k;

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, menu_color, menu_color, field_color);
   sed_SetPosition(sed, 1, 10);

   sed_Push(sed);
   sed_Repaint(sed);
   show_key_help(&key_menu, &key_sed, no_default_help_string);

   do
      {
      ret = sed_Go(sed);

/* ========================================================================== */
/*  If the parameters entered check out OK, recalc a k value.                 */
/* ========================================================================== */

      if ((ret) && (check_nonzero(k_calc_info, model_info->l0)))
         {
         sed_DoFieldFexit(sed, sed_GetFieldNo(sed));
         new_k = calculate_k(k_calc_info, model_info);
         }

      switch (ret)
         {
         case FN1:
            compile_help(CALCULATE_K_MENU);
            sed_SetBaton(sed, -1);
            break;

         case FN2:
```

```
            switch (sed_GetFieldNo(sed))
               {
               case 0:
                  k_calc_info->well_depth = DEFAULT_WELL_DEPTH;
                  break;

               case 1:
                  k_calc_info->radius = DEFAULT_INFLUENCE_RAD;
                  break;

               case 2:
                  k_calc_info->density = DEFAULT_REFUSE_DENSITY;
                  break;

               case 3:
                  k_calc_info->fraction = DEFAULT_DECOMP_FRACTION;
                  break;

               case 4:
                  if (config.default_mode == AP_MODE)
                     model_info->l0 = DEFAULT_AP42_L0;
                  else model_info->l0 = DEFAULT_REGULATORY_L0;
                  break;

               case 5:
                  k_calc_info->flow_rate = DEFAULT_FLOW_RATE;
                  break;

               case 6:
                  k_calc_info->refuse_age = DEFAULT_REFUSE_AGE;
                  break;
               }
         }

      sed_RepaintFields(sed);
      }
   while ((ret) && (ret != FN10));

   sed_DoFieldSexit(sed, sed_GetFieldNo(sed));

   if (ret == FN10) model_info->k = new_k;

   destroy_key_help(key_menu, key_sed);
   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);

   susdbl_funcs.fkey = old_fkey;

   return(ret);
}


boolean check_nonzero(k_calc_info, l0)
struct k_calc_parms *k_calc_info;
double l0;
{
boolean ret = TRUE;

   if (k_calc_info->well_depth <= 0.0)
      ret = FALSE;

   if (k_calc_info->radius <= 0.0)
      ret = FALSE;
```



```
   if (k_calc_info->density <= 0.0)
      ret = FALSE;

   if (k_calc_info->fraction <= 0.0)
      ret = FALSE;

   if (k_calc_info->flow_rate <= 0.0)
      ret = FALSE;

   if (k_calc_info->refuse_age <= 0.0)
      ret = FALSE;

   if (l0 <= 0.0)
      ret = FALSE;

   return(ret);
}


void recalc_fkey(sed)
sed_type sed;
{
int scancode, key;

   scancode = kb_Read();

   if (special_key(sed, scancode)) return;
   if (recalc_inter_field(sed, scancode)) return;
   if (inter_page(sed, scancode))  return;

   if ((ascii(scancode)) && (sed_GetRecordPos(sed) == 0) && (! kb_Insert()))
      {
        sed_SetRecord(sed, "\0", sed_GetFieldNo(sed));
      sed_UpdateCurrField(sed);
      }

   switch(scancode)
      {
      case RIGHT:
         sed_IncChar(sed);
         break;

      case LEFT:
         sed_DecChar(sed);
        break;

      case HOME:
         sed_GoHome(sed);
               break;

      case END:
         sed_GoEnd(sed);
               break;

      case INS:
         if (kb_Insert())
            vid_SetCursorType(CURSOR_HALF);
         else
            vid_SetCursorType(CURSOR_NORMAL);
         break;

      case BACKSPACE:
         if (sed_DecChar(sed)) sed_PullRight(sed);
         break;

      case DEL:
```

```
         sed_PullRight(sed);
         break;

      default:
         key = ascii(scancode);
         if (isdigit(key) || key == '-' || key == '.' || toupper(key) == 'E')
            {
            if (kb_Insert())
               sed_PushRight(sed, key);
            else
               sed_Overwrite(sed, key);

            sed_IncChar(sed);
            }

         break;
      }
}


boolean recalc_inter_field(sed, scancode)
sed_type sed;
int scancode;
{
   switch (scancode)
      {
      case ESC:
         sed_SetBaton(sed, 0);
         sed_ToggleExit(sed);
         return(TRUE);

      case ENTER:
         sed_SetBaton(sed, sed_GetFieldNo(sed)+1);
         sed_IncField(sed);
         sed_ToggleExit(sed);
         return(TRUE);

      case UP:
         sed_DecField(sed);
         sed_SetBaton(sed, sed_GetFieldNo(sed)+1);
         sed_ToggleExit(sed);
         return(TRUE);

      case DOWN:
         sed_IncField(sed);
         sed_SetBaton(sed, sed_GetFieldNo(sed)+1);
         sed_ToggleExit(sed);
         return(TRUE);

      default:
         break;
      }

   return(FALSE);
}
```

```
/* ========================================================================== */
/*                                KCALCMN.C                                   */
/* ========================================================================== */
/*  This file contains the code for calculating a k value from input          */
/*  parameters obtained by testing.                                           */
/* ========================================================================== */
/*                            Created under Contract                          */
/*                                    to                                      */
/*                              EPA/AEERL/GECD/EMB                            */
/*                                                                            */
/*                                    by                                      */
/*                                                                            */
/*                              Radian Corporation                            */
/*                           Research Triangle Park, NC                       */
/* ========================================================================== */
/* ========================================================================== */
/*                              Revision History                              */
/* ========================================================================== */
/*    07/03/90    rlb       Initial Implementation                            */
/* ========================================================================== */

#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include "landfill.h"

#include "constant.h"

/* ========================================================================== */
/*          Determine the calculation limitations of the system.              */
/* ========================================================================== */

#define FINE_ITERATIONS 10000
#define COARSE_ITERATIONS 50
#define TOLERANCE 0.000001

/* ========================================================================== */
/*  This function bears a little description.  The function cannot be         */
/*  solved analytically for k.  Rather, for a given set of parameters,        */
/*  k must be approximated.                                                   */
/*                                                                            */
/*  The function can be shown to have two roots.  One root lies in the        */
/*  interval 0 and 1/refuse_age (the maximum of the function) and the         */
/*  other between 1/refuse_age and infinity.  The root of interest lies       */
/*  closest to the origin.  Therefore, in order to insure the correct         */
/*  root is found, this function implements two different approximation       */
/*  methods for the root:  BISECTION (used to adjust the approximation        */
/*  corsely) and NEWTON-RAPHSON (used to make fine adjustments to the         */
/*  approximation.                                                            */
/*                                                                            */
/*  BISECTION attempts to find the root on an interval by adding or           */
/*  subtracting ever decreasing corrections to a guess lying on the           */
/*  interval in an attempt to zero in on the root.                            */
/*                                                                            */
/*  After a rough but close estimation of the root, NEWTON-RAPHSON makes      */
/*  changes to the estimation be altering the value based on the              */
/*  derivative of the function.  Given a value close to the root, this        */
/*  method requires only a few iterations to achieve acceptable accuracy.    */
/* ========================================================================== */

double calculate_k(k_calc_info, model_info)
struct k_calc_parms *k_calc_info;
struct model_parms *model_info;
[
double affected_depth, refuse_volume, affected_refuse_mass;
double adjusted_l0, old_k, new_k, delta_k, correction;
```



```
double old_res, new_res;
double max_value;

int count = 0;

double landfill_function(), landfill_derivative();

/* ========================================================================== */
/*  The affected depth cannot exceed the landfill depth.  This assumes        */
/*  the well depth is 0.75 of the landfill depth in order to determine if     */
/*  this is so.                                                               */
/* ========================================================================== */

   affected_depth = k_calc_info->well_depth + k_calc_info->radius;

   if (affected_depth > k_calc_info->well_depth / 0.75)
      affected_depth = k_calc_info->well_depth / 0.75;

/* ========================================================================== */
/*            Calculate parameters required for the equation.                 */
/* ========================================================================== */

   refuse_volume = PI * affected_depth * k_calc_info->radius *
                        k_calc_info->radius;

   affected_refuse_mass = refuse_volume * k_calc_info->density;

   adjusted_l0 = k_calc_info->fraction * model_info->l0;

/* ========================================================================== */
/*  If the landfill function is less than zero at the local maximum           */
/*  (which can be shown to be 1/refuse_age), no solution can be found.        */
/*  Automatically return a -99999.                                            */
/* ========================================================================== */

  if ((max_value = landfill_function((1 / k_calc_info->refuse_age),
                        k_calc_info->refuse_age,
                        k_calc_info->flow_rate, adjusted_l0,
                        affected_refuse_mass)) < 0) return(-99999);

/* ========================================================================== */
/*  Just in case the parameters are already such that the maximum is the      */
/*  zero of the function, check to see and return it.                         */
/* ========================================================================== */


   if (max_value < TOLERANCE) return(max_value);

/* ========================================================================== */
/*  Set up for the BISECTION algorithm.  The initial correction value is      */
/*  set to 1/refuse_age (the maximum of the function).  Two values for k      */
/*  are estimated.  For the zero case, the landfill function will return      */
/*  its most negative value (on the interval of interest).  The direction     */
/*  of search is set initially to increasing k value.                         */
/* ========================================================================== */

   delta_k = 1 / (2 * k_calc_info->refuse_age);

   old_k = 0.0;

   old_res = landfill_function(old_k, k_calc_info->refuse_age,
                        k_calc_info->flow_rate, adjusted_l0,
                           affected_refuse_mass);

   old_k = delta_k;
```

```
   while (count++ < COARSE_ITERATIONS)
      {
/* ================================================================== */
/*      Determine the function value using the corrected value of k.  */
/* ================================================================== */

      new_res = landfill_function(old_k, k_calc_info->refuse_age,
                        k_calc_info->flow_rate, adjusted_l0,
                           affected_refuse_mass);

/* ================================================================== */
/*  If we manage to determine the k value accurately enough here, return  */
/*  the value and ignore the rest of the function.                    */
/* ================================================================== */

      if (fabs(new_res) < TOLERANCE) return(old_k);

/* ================================================================== */
/*  As soon as the new value of the landfill function changes sign from   */
/*  old value, reverse the correction direction and half the correction   */
/*  factor.                                                           */
/* ================================================================== */

      if (old_res * new_res < 0.0) delta_k = -(delta_k) / 2.0;

/* ================================================================== */
/*  Correct the old value of k and set the new result to the old result   */
/* ================================================================== */

      old_k += delta_k;
      old_res = new_res;
      }

/* ================================================================== */
/*  Here begins the NEWTON-RAPHSON step.  Because BISECTION is slow, we   */
/*  can't wait until a root is found with an acceptable tolerance.  Use   */
/*  this portion to refine the root.                                  */
/* ================================================================== */

   while (count++ < FINE_ITERATIONS)
      {
      correction = landfill_function(old_k, k_calc_info->refuse_age,
                        k_calc_info->flow_rate, adjusted_l0,
                           affected_refuse_mass);

      correction /= landfill_derivative(old_k, k_calc_info->refuse_age);

      new_k = old_k - correction;

      if ((old_k - new_k) < TOLERANCE) return(new_k);

      old_k = new_k;
      }

/* ================================================================== */
/*       Returns a -99999 if no value was found after both methods.   */
/* ================================================================== */

   return(-99999);
}


double landfill_function(k, refuse_age, flow_rate, adjusted_l0,
                         affected_refuse_mass)
double k, refuse_age, flow_rate, adjusted_l0, affected_refuse_mass;
{
```



```
double term1, term2;

   term1 = k * exp(-k * refuse_age);

   term2 = 525960.0 * (flow_rate / (2 * adjusted_l0 * affected_refuse_mass));

   return(term1 - term2);
}


double landfill_derivative(k, refuse_age)
double k, refuse_age;
{
double term1, term2;

   term1 = exp(-k * refuse_age);

   term2 = (1 - refuse_age * k);

   return(term1 * term2);
}
```

```
/* ========================================================================== */
/*                                  KEDIT.C                                   */
/* ========================================================================== */
/*     This file contains the code for displaying the k entry screen.         */
/* ========================================================================== */
/*                             Created under Contract                         */
/*                                       to                                   */
/*                               EPA/AEERL/GECD/EMB                           */
/*                                                                            */
/*                                       by                                   */
/*                                                                            */
/*                               Radian Corporation                           */
/*                            Research Triangle Park, NC                      */
/* ========================================================================== */
/* ========================================================================== */
/*                              Revision History                              */
/* ========================================================================== */
/*    07/03/90   rlb       Initial Implementation                             */
/* ========================================================================== */

#include <stdio.h>
#include <cscape.h>
#include <scancode.h>
#include "landfill.h"
#include "landuser.h"
#include "defaults.h"
#include "helpasgn.h"
#include "colors.h"

int edit_k_value(k)
double *k;
{
menu_type   menu, key_menu;
sed_type    sed, key_sed;
int         ret;
double new_k;

   new_k = *k;

   reset_insert_status();

   menu = menu_Open();

   buildbox(menu, menu_color, 7, 42);
   menu_Printf(menu, "@p[1,5]Methane Generation Rate Constant");
   menu_Printf(menu, "@p[4,4]k :");
   menu_Printf(menu, "@p[4,31]yr^-1");
   menu_Printf(menu, "@p[4,9]@f[@[20,#]]",
     &new_k, &susdbl_funcs);

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, menu_color, menu_color, field_color);
   sed_SetPosition(sed, 9, 19);

   sed_Push(sed);
   sed_Repaint(sed);
   show_key_help(&key_menu, &key_sed, generic_help_string);

   do
      {
      ret = sed_Go(sed);

      switch (ret)
         {
```



```
         case FN1:
            compile_help(EDIT_K_VALUE);
            sed_SetBaton(sed, -1);
            break;

         case FN2:
            if (config.default_mode == AP_MODE)
               new_k = DEFAULT_AP42_K;
            else new_k = DEFAULT_REGULATORY_K;
            sed_RepaintFields(sed);
            break;
         }
      }
   while ((ret != FN10) && (ret));

   if (ret == FN10) *k = new_k;

   destroy_key_help(key_menu, key_sed);
   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);
   return(ret);
}
```

```
/* ========================================================================== */
/*                                KEYHELP.C                                   */
/* ========================================================================== */
/*  This file contains the code for displaying the key help at the bottom     */
/*  of the screen.                                                            */
/* ========================================================================== */
/*                            Created under Contract                          */
/*                                      to                                    */
/*                              EPA/AEERL/GECD/EMB                            */
/*                                                                            */
/*                                      by                                    */
/*                                                                            */
/*                              Radian Corporation                            */
/*                           Research Triangle Park, NC                       */
/* ========================================================================== */
/* ========================================================================== */
/*                              Revision History                              */
/* ========================================================================== */
/*    07/03/90    rlb        Initial Implementation                           */
/* ========================================================================== */

#include <stdio.h>
#include <cscape.h>
#include <string.h>
#include "keyhelp.h"
#include "colors.h"

int show_key_help(menu, sed, help_string)
menu_type   *menu;
sed_type    *sed;
char *help_string;
{
char help_line[85];

   strcenter(strcpy(help_line, help_string), 80);

   *menu = menu_Open();

   menu_Printf(*menu, "@p[0,0]@c[%c]%s@c[\x07]", keyhelp_color, help_line);

   menu_Flush(*menu);

   *sed = sed_Open(*menu);
   sed_SetColors(*sed, keyhelp_color, keyhelp_color, keyhelp_color);
   sed_SetPosition(*sed, 24, 0);

   sed_Push(*sed);
   sed_Repaint(*sed);

   return(0);
}



int destroy_key_help(menu, sed)
menu_type menu;
sed_type sed;
{

   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);
   return(0);
}
```

```
/* ========================================================================== */
/*                                  MAIN.C                                    */
/* ========================================================================== */
/*  This file contains the main controlling module for the program as         */
/*  well as the functions that initialize studies.                            */
/* ========================================================================== */
/*                            Created under Contract                          */
/*                                    to                                      */
/*                              EPA/AEERL/GECD/EMB                            */
/*                                                                            */
/*                                    by                                      */
/*                                                                            */
/*                              Radian Corporation                            */
/*                           Research Triangle Park, NC                       */
/* ========================================================================== */
/* ========================================================================== */
/*                             Revision History                               */
/* ========================================================================== */
/*    07/03/90   rlb       Initial Implementation.                            */
/*    01/17/94   rlb       Changed study retrieval routine to support         */
/*                            the codisposal/no codisposal factors.           */
/*    01/18/94   rlb       Changed init_new_study include_air_toxics flag     */
/*                            to 0 to reflect default under model             */
/*                            revision.                                       */
/*    01/29/94   rlb       Changed to permit the defaults selection.          */
/* ========================================================================== */

#include <stdio.h>
#include <ctype.h>
#include <cscape.h>
#include <dos.h>
#include <bios.h>
#include <errno.h>
#include <malloc.h>
#include <memory.h>
#include <string.h>
#include <direct.h>
#include "landfill.h"
#include "defaults.h"
#include "colors.h"
#include "colordfs.h"

extern void credits(void);
extern int study_type(void);
extern boolean edit_main(struct landfill_parms *, struct model_parms *,
                         struct k_calc_parms *, boolean *);
extern int main_menu(int);
extern int get_study_name(char *, int);
extern void no_study_selected();

static char prn_table[][6] = {"LPT1", "LPT2", "COM1", "COM2"};


/* ========================================================================== */
/*                    Declare the system parameter global                     */
/* ========================================================================== */

struct system_parms config;


#define NEW_STUDY 1
#define OLD_STUDY 2

main(argc, argv)
int argc;
char *argv[];
```



```
{
char parm_string[256];
char study_name[9];
int i, choice = 1, tab_choice = 1, type = 0, old_type;
int old_cursor;
unsigned int old_row, old_col, mode, mem, code, free;
struct k_calc_parms k_calc_info;
extern struct system_parms config;

static struct landfill_parms landfill_info = {0, 0, 0, 0, 0, 0, 0, NULL};
static struct model_results results = { 0, 0, NULL, NULL, NULL, NULL, NULL,
                                        NULL };
static struct model_parms model_info = {DEFAULT_REGULATORY_L0,
                                        DEFAULT_REGULATORY_K,
                                        DEFAULT_REGULATORY_NMOC,
                                        DEFAULT_CH4, DEFAULT_CO2, 0, NULL, 0};


boolean calc_possible = FALSE, calc_done = FALSE, exit_requested = FALSE,
        study_changed = FALSE;

void init_new_study();
int init_old_study();

/* ========================================================================== */
/*  Set the memory allocation block size to the lowest possible (since I      */
/*  am allocating memory in blocks myself, I think I'm OK here).              */
/* ========================================================================== */

   _amblksiz = 1;

/* ========================================================================== */
/*  To the best of my knowledge, the CSCAPE functions do not provide a        */
/*  means of resetting the insert status bit on entry.  This gets to be       */
/*  a real pain because the cursor keeps coming up in insert mode that        */
/*  is a result of pressing the insert key external to the program.  As       */
/*  as I hate to do it, I have implemented this kludge to handle the          */
/*  problem.  This function (see MISC_UTL.C) resets the bit indicating        */
/*  insert mode by reading, altering, and writing directly the ROM BIOS       */
/*  data area.                                                                */
/* ========================================================================== */

/*   reset_insert_status(); */

   vid_GetCursorPos(&old_row, &old_col);

   vid_Clear('\x07');

   old_cursor = vid_GetCursorType();
   vid_SetCursorType(CURSOR_NONE);

   *study_name = '\0';

/* ========================================================================== */
/*  Determine the current text mode.  If the screen is mono, remap the        */
/*  global colors to black and white.                                         */
/* ========================================================================== */

   mode = vid_GetMode();
   if (mode < 2) vid_SetMode(mode + 2);

   if ((mode == 7) || (mode == 2))
      {
      menu_color = (BLACK << 4) | WHITE;
      warning_color = (BLACK << 4) | WHITE;
      error_color = (BLACK << 4) | WHITE;
```

```
      info_color = (BLACK << 4) | WHITE;
      keyhelp_color = (WHITE << 4) | BLACK;
      field_color = (WHITE << 4) | BLACK;
      }

/* ========================================================================= */
/*  Check to see if the user has enough available memory to execute this     */
/*  application.  This is done by requesting all of the memory contained     */
/*  in the system.  This will fail but the dos service will return in        */
/*  free variable the number of paragraphs (16 bytes) available.             */
/* ========================================================================= */

#ifndef NOMEMCHK

   errno = 0;
   code = _dos_allocmem((unsigned) _bios_memsize() * 64 - 1, &free);
   if (errno == ENOMEM)
      {
      if (free < 8200)
         {
         show_mem_limitation();

         vid_SetCursorType(old_cursor);
         vid_SetCursorPos(old_row, old_col);

         exit(0);
         }
      }
   else
      _dos_freemem(free);

#endif

/* ========================================================================= */
/*                            Display title screen.                          */
/* ========================================================================= */

   credits();

/* ========================================================================= */
/*             Setup system directory and printer information                */
/* ========================================================================= */

   set_system_parms(argv[0]);

/* ========================================================================= */
/* Use the just determined default mode to set the operating parameters      */
/* for the program.                                                          */
/* ========================================================================= */

   if (config.default_mode == AP_MODE)
      {
      model_info.l0 = DEFAULT_AP42_L0;
      model_info.k = DEFAULT_AP42_K;

      if (model_info.include_air_toxics == CODISPOSAL)
         model_info.nmoc = DEFAULT_AP42_NMOC_CODIS;
      else model_info.nmoc = DEFAULT_AP42_NMOC_NOCO;
      }

/* ========================================================================= */
/*                            Process Menu Entry                             */
/* ========================================================================= */

   do
      {
```



```
      choice = main_menu(choice - 1);

      switch (choice)
         {
         case 0:
            if (exit_program() == 1) exit_requested = TRUE;
            else choice = 1;
            break;

         case 1:
            old_type = type;

            if ((type) && (study_changed))
               if (display_save_menu() == 1)
                  save_study_info(study_name, landfill_info, model_info,
                                  calc_possible);

            switch(type = study_type())
               {
               case 1:
                  if (get_study_name(study_name, type))
                     {
                     init_new_study(&landfill_info, &model_info,
                                    &k_calc_info);
                     calc_possible = calc_done = FALSE;
                     study_changed = TRUE;
                     }
                  else type = old_type;

                  break;

               case 2:
                  if (get_study_name(study_name, type))
                     {
                     if (init_old_study(study_name, &landfill_info,
                                        &model_info, &k_calc_info,
                                        &calc_possible)) type = 0;
                     else study_changed = calc_done = FALSE;
                     }
                  else type = old_type;

                  break;

               default:
                  type = old_type;
               }

            break;

         case 2:
            if (type) study_changed =
                        edit_main(&landfill_info, &model_info,
                                  &k_calc_info, &calc_possible);
            else no_study_selected();
            break;

         case 3:
            if ((type) && (calc_possible))
               calculation_main(&landfill_info, &model_info, &results,
                                &calc_done);
            else if (!type) no_study_selected();
            else if (! calc_possible) warn_no_data();
            break;

         case 4:
            if ((type) && (calc_done))
```

```
            report_model_calc(&landfill_info, &model_info, &results);
         else if (! type) no_study_selected();
         else if (! calc_done) warn_no_calc();
         break;

      case 5:
         configure_system();

         if (config.default_mode == AP_MODE)
            {
            model_info.l0 = DEFAULT_AP42_L0;
            model_info.k = DEFAULT_AP42_K;

            if (model_info.include_air_toxics == CODISPOSAL)
               model_info.nmoc = DEFAULT_AP42_NMOC_CODIS;
            else model_info.nmoc = DEFAULT_AP42_NMOC_NOCO;
            }
         else
            {
            model_info.l0 = DEFAULT_REGULATORY_L0;
            model_info.k = DEFAULT_REGULATORY_K;

            model_info.nmoc = DEFAULT_REGULATORY_NMOC;
            }

         break;

      case 6:
         exit_requested = TRUE;
         break;
      }
   }
while (! exit_requested);

if ((type) && (exit_requested) && (study_changed))
   if (display_save_menu() == 1)
      save_study_info(study_name, landfill_info, model_info,
                      calc_possible);

vid_SetCursorType(old_cursor);

vid_SetCursorPos(old_row, old_col);
}


int report_model_calc(landfill, model, results)
struct model_parms *model;
struct landfill_parms *landfill;
struct model_results *results;
{
static int disp_choice = 1, tab_choice = 1;
int report_choice;
char outfile[15];

   *outfile = '\0';

   do
      {
      disp_choice = display_results_menu(disp_choice - 1);

      switch (disp_choice)
         {
         case 1:
            do
               {
```



```
               tab_choice = tabular_report_menu(tab_choice - 1);

               switch (tab_choice)
                  {
                  case 1:
                     if (report_choice =
                        report_selector(model->air_toxics,
                           model->number_air_toxics, NULL))
                        show_results(model, landfill, results,
                                     report_choice);

                     break;

                  case 2:
                     if (report_choice = report_selector(model->air_toxics,
                           model->number_air_toxics, "ALL"))
                        {
                        if (report_choice == model->number_air_toxics + 4)
                           print_results(NULL, prn_table[config.print_device],
                                     model, landfill, results,
                                     0);
                        else
                           print_results(NULL, prn_table[config.print_device],
                                     model, landfill, results,
                                     report_choice);
                        }
                     break;

                  case 3:
                     if (get_output_filename(outfile))
                        {
                        if (report_choice = report_selector(model->air_toxics,
                           model->number_air_toxics, "ALL"))
                              {
                              if (report_choice == model->number_air_toxics + 4)
                                 print_results(config.working_dir, outfile,
                                          model, landfill, results,
                                          0);
                              else
                                 print_results(config.working_dir, outfile,
                                          model, landfill, results,
                                          report_choice);
                              }
                        }


                     break;
                  }
               }
            while ((tab_choice > 0) && (tab_choice < 4));

            if (tab_choice == 0) tab_choice = 1;
            break;

         case 2:
            if (report_choice = report_selector(model->air_toxics,
                           model->number_air_toxics, NULL))
               switch (report_choice)
                  {
                  case 0:
                     break;

                  case 1:
                     display_onscreen_graph(results->year_list,
                                            results->annual_ch4,
                                            landfill->current_year,
```

```
                                        results->year_count,
                                        "Methane", "Cft/yr");
                        break;

                     case 2:
                        display_onscreen_graph(results->year_list,
                                        results->annual_co2,
                                        landfill->current_year,
                                        results->year_count,
                                        "Carbon Dioxide", "Cft/yr");
                        break;

                     case 3:
                        display_onscreen_graph(results->year_list,
                                        results->annual_nmoc,
                                        landfill->current_year,
                                        results->year_count,
                                        "NMOC", "Mg/yr");

                        break;

                     default:
                        display_onscreen_graph(results->year_list,
                                        results->annual_air_toxics[report
_choice - 4],
                                        landfill->current_year,
                                        results->year_count,
                                        model->air_toxics[report_choice -
 4].chemical_name,
                                        "Cft/yr");
                        break;
                     }

               break;
            }
         }
      while ((disp_choice > 0) && (disp_choice < 3));

      if (disp_choice == 0) disp_choice = 1;
}



int set_system_parms(path)
char *path;
{
FILE *config_file;
int i, len;
boolean trap = FALSE;
char string[256];

extern struct system_parms config;

/* ================================================================== */
/*  This looks kind of complicated but it's not.  The intent here is to  */
/*  determine the landfill system directory from the arguments passed by  */
/*  DOS to the program.  The first argument (0) is the full path to the  */
/*  program being executed.  This section starts at the end of the arg  */
/*  string and search backward, looking for the backslash.  For each  */
/*  character encountered before the backslash, a null is put into the  */
/*  target string.  As soon as the backslash is found, it is copied to  */
/*  the target string and all other characters to the first one are also  */
/*  placed in the string.                                             */
/* ================================================================== */

   for (i = strlen(path) - 1; i >= 0 ; i--)
```



```
      {
      if (trap)
         config.landfill_dir[i] = path[i];
      else
         {
         if (path[i] == '\\')
            {
            config.landfill_dir[i] = path[i];
            trap = TRUE;
            }
         else
            config.landfill_dir[i] = '\0';
         }
      }

   strcpy(string, config.landfill_dir);
   strcat(string, "LANDFILL.CFG");

   config_file = fopen(string, "r");

   if (config_file == NULL)
      {
      config_file = fopen(string, "w");

/* ================================================================== */
/*   Get the current working directory and place it in a place so it can  */
/*   be changed if the user wants.                                    */
/* ================================================================== */

      len = strlen(getcwd(config.working_dir, 255));

      if (*(config.working_dir + len - 1) == '\\')
         *(config.working_dir + len - 1) = '\0';

/* ================================================================== */
/*          Set default print device to 0 for PRN (1 = AUX)           */
/* ================================================================== */

      config.print_device = 0;
      config.default_mode = 0;
      config.years_passed_closure = 50;

      fprintf(config_file, "%s\n", config.working_dir);
      fprintf(config_file, "%d\n", config.print_device);
      fprintf(config_file, "%d\n", config.default_mode);
      fprintf(config_file, "%d\n", config.years_passed_closure);
      }
   else
      {
/* ================================================================== */
/*   Get the current working directory and place it in a place so it can  */
/*   be changed if the user wants.                                    */
/* ================================================================== */

      fscanf(config_file, "%s", config.working_dir);

/*      len = strlen(getcwd(config.working_dir, 255));

      if (*(config.working_dir + len - 1) == '\\')
         *(config.working_dir + len - 1) = '\0';
*/

      fscanf(config_file, "%d", &(config.print_device));
      fscanf(config_file, "%d", &(config.default_mode));
      fscanf(config_file, "%d", &(config.years_passed_closure));
      }
```

```
   fclose(config_file);

   return(0);
}



void init_new_study(landfill_info, model_info, k_calc_info)
struct landfill_parms *landfill_info;
struct model_parms *model_info;
struct k_calc_parms *k_calc_info;
{
struct dosdate_t date;
extern struct system_parms config;

void initialize_air_toxics(char *, char *, struct air_toxic_structure **,
                           int *);

   _dos_getdate(&date);

   landfill_info->year_opened = (int) (date.year - DEFAULT_YEARS_PRIOR);
   landfill_info->current_year = (int) date.year;

   landfill_info->capacity = 0.0;

   landfill_info->refuse_in_place =
      (double *) calloc(DEFAULT_YEARS_PRIOR, sizeof(double));

   if (landfill_info->refuse_in_place == NULL)
      show_mem_error("init_new_study", "refuse_in_place");

   landfill_info->number_of_periods = DEFAULT_YEARS_PRIOR;

   if (config.default_mode == REGULATORY_MODE)
      {
      model_info->l0 = DEFAULT_REGULATORY_L0;
      model_info->k = DEFAULT_REGULATORY_K;

      model_info->nmoc = DEFAULT_REGULATORY_NMOC;
      }
   else
      {
      model_info->l0 = DEFAULT_AP42_L0;
      model_info->k = DEFAULT_AP42_K;
      model_info->nmoc = DEFAULT_AP42_NMOC_NOCO;
      }

   model_info->ch4 = DEFAULT_CH4;
   model_info->co2 = DEFAULT_CO2;
   model_info->include_air_toxics = NOCODISPOSAL;

   initialize_air_toxics(config.landfill_dir, "airtoxic.txt",
                         &(model_info->air_toxics),
                         &(model_info->number_air_toxics));

   k_calc_info->well_depth = DEFAULT_WELL_DEPTH;
   k_calc_info->radius = DEFAULT_INFLUENCE_RAD;
   k_calc_info->density = DEFAULT_REFUSE_DENSITY;
   k_calc_info->fraction = DEFAULT_DECOMP_FRACTION;
   k_calc_info->flow_rate = DEFAULT_FLOW_RATE;
   k_calc_info->refuse_age = DEFAULT_REFUSE_AGE;
}


int init_old_study(study_name, landfill_info, model_info, k_calc_info,
```



```
                   calc_possible)
char *study_name;
struct landfill_parms *landfill_info;
struct model_parms *model_info;
struct k_calc_parms *k_calc_info;
boolean *calc_possible;
{
int i;
char string[256], filename[15];
extern struct system_parms config;
FILE *parameters;

void initialize_air_toxics(char *, char *, struct air_toxic_structure **,
                           int *);

   strcpy(string, config.working_dir);
   strcat(string, "\\");
   strcat(string, study_name);
   strcat(string, ".prm");

   parameters = fopen(string, "r");

   if (parameters == NULL)
      {
      show_file_error();
      return(1);
      }

   fscanf(parameters, "%lf %lf %lf %lf %lf %d",  &(model_info->l0),
               &(model_info->k), &(model_info->nmoc),
               &(model_info->ch4), &(model_info->co2),
               &(model_info->include_air_toxics));

   fscanf(parameters, "%d %d %d", &(landfill_info->year_opened),
                                  &(landfill_info->current_year),
                                  &(landfill_info->year_closed));

   fscanf(parameters, "%lf %lf %lf", &(landfill_info->capacity),
                                  &(landfill_info->avg_acceptance_rate),
                                  &(landfill_info->last_rate));

   fscanf(parameters, "%d", (int *) calc_possible);

   fscanf(parameters, "%d", &(landfill_info->number_of_periods));

   if (landfill_info->refuse_in_place != NULL)
      free(landfill_info->refuse_in_place);

   landfill_info->refuse_in_place =
      (double *) calloc((size_t) landfill_info->number_of_periods,
               sizeof(double));

   if (landfill_info->refuse_in_place == NULL)
      show_mem_error("init_new_study", "refuse_in_place");

   for (i = 0 ; i < landfill_info->number_of_periods ; i++)
      fscanf(parameters, "%lf", &(landfill_info->refuse_in_place[i]));

   fclose(parameters);

   strcpy(string, config.working_dir);
   strcat(string, "\\");

   strcpy(filename, study_name);
   strcat(filename, ".atx");
```

Appendix A

DOS Version NMake File

```
#               Make file for the LANDFILL PROGRAM - 04/04/90
#
#                       Written by RADIAN CORPORATION
#                        Research Triangle Park, NC
#
# Modified by RLB (Radian) 12/93 - Included support for a portable
#           working environment during the development and allowing
#           for different origins of the libraries and includes.

# Set the macro definitions for the executable and the link
# libraries here.

exec = landfill.exe
map = xxx.map
debug = /Z7 /DNOMEMCHK

# Different directories for the compilation headers and libraries
# depending on the system.

!IF "$(WORKSTATION)" == "PORTABLE"

BIN=d:\msvc\bin
INCLUDE=d:\msvc\include
LIB=d:\msvc\lib

cscape_include = d:\proglang\old_cscp\include\
cscape_lib = d:\proglang\old_cscp\lib\

!ELSE

BIN=F:\msvc\bin
INCLUDE = F:\msvc\include
LIB=F:\msvc\lib

cscape_include = c:\proglang\old_cscp\include\
cscape_lib = c:\proglang\old_cscp\lib\

!ENDIF

# Compilation command with options - /Zi /Zd are debug options
cc = $(BIN)\cl /c /AL /I$(cscape_include) /I$(INCLUDE)


# Link options - /CO is the debug option
link_opts = /STACK:16000 /SE:256

link = $(BIN)\link $(link_opts)


$(exec) : main.obj editmenu.obj mainmenu.obj new-old.obj credits.obj \
          speckey.obj fnsusdbl.obj misc_utl.obj landfill.h landuser.h \
          methrate.obj methcap.obj chemcomp.obj editmain.obj airtoxmn.obj \
          periods.obj fnsusint.obj kcalcmn.obj kcalcmen.obj kedit.obj \
          disprslt.obj tblemenu.obj studname.obj closure.obj vferror.obj \
          fndosstr.obj warnings.obj calcmain.obj scr_view.obj help_scr.obj \
          summary.obj selector.obj config.obj exit.obj savemenu.obj  \
          savestud.obj fnticker.obj strmview.obj fndosfil.obj getout.obj \
          do_graph.obj keyhelp.obj buildbox.obj flselect.obj findfls.obj \
          fnselect.obj
          @SET LIB=$(LIB)
          @echo main editmenu mainmenu new-old credits + > landfill.lnk
          @echo speckey fnsusdbl misc_utl methrate methcap + >> landfill.lnk
          @echo chemcomp editmain airtoxmn periods fnsusint + >> landfill.lnk
          @echo kcalcmn kcalcmen kedit disprslt tblemenu studname + >> landfill.
lnk
          @echo closure vferror fndosstr warnings calcmain scr_view + >> landfil
```



```
l.lnk
          @echo help_scr summary selector config exit savemenu + >> landfill.lnk
          @echo savestud fnticker strmview fndosfil getout do_graph + >> landfil
l.lnk
          @echo keyhelp buildbox flselect findfls fnselect >> landfill.lnk
          @echo $(exec),$(map) >> landfill.lnk
          @echo $(cscape_lib)m5lcscap.lib pgchart.lib $(link_opts) ; >> landfill
.lnk

          @$(link) @landfill.lnk

          @del landfill.lnk

main.obj : main.c landfill.h defaults.h

do_graph.obj : do_graph.c landfill.h

keyhelp.obj : keyhelp.c keyhelp.h

config.obj : config.c landfill.h helpasgn.h

editmenu.obj : editmenu.c landfill.h

editmain.obj : editmain.c landfill.h

calcmain.obj : calcmain.c landfill.h defaults.h

summary.obj : summary.c landfill.h landuser.h defaults.h

scr_view.obj : scr_view.c landfill.h defaults.h

strmview.obj : strmview.c landfill.h defaults.h

selector.obj : selector.c landfill.h

mainmenu.obj : mainmenu.c

exit.obj : exit.c

savemenu.obj : savemenu.c

savestud.obj : savestud.c

buildbox.obj : buildbox.c

help_scr.obj : help_scr.c

studname.obj : studname.c

disprslt.obj : disprslt.c

tblemenu.obj : tblemenu.c

speckey.obj : speckey.c

fnsusdbl.obj : fnsusdbl.c

fnticker.obj : fnticker.c

fnsusint.obj : fnsusint.c

fndosstr.obj : fndosstr.c

fndosfil.obj : fndosfil.c

fnselect.obj : fnselect.c
```

```
getout.obj : getout.c landuser.h

misc_utl.obj : misc_utl.c

credits.obj : credits.c landfill.h

findfls.obj : findfls.c landfill.h

flselect.obj : flselect.c landfill.h

new-old.obj : new-old.c

methrate.obj : methrate.c defaults.h

methcap.obj : methcap.c defaults.h

kedit.obj : kedit.c defaults.h

chemcomp.obj : chemcomp.c landfill.h defaults.h

airtoxmn.obj : airtoxmn.c landfill.h landuser.h defaults.h

periods.obj : periods.c landfill.h landuser.h defaults.h

closure.obj : closure.c landfill.h landuser.h defaults.h

kcalcmen.obj : kcalcmen.c landfill.h landuser.h defaults.h

kcalcmn.obj : kcalcmn.c landfill.h constant.h defaults.h

vferror.obj : vferror.c vfmsgs.h

warnings.obj : warnings.c
```

Appendix B

DOS Version Code Listings

```
/* ========================================================================== */
/*                                  COLORDFS.H                                */
/* ========================================================================== */
/*  This header file contains the global color information for the menus      */
/* ========================================================================== */
/*                             Created under Contract                         */
/*                                       to                                   */
/*                               EPA/AEERL/GECD/EMB                           */
/*                                                                            */
/*                                       by                                   */
/*                                                                            */
/*                               Radian Corporation                           */
/*                            Research Triangle Park, NC                      */
/* ========================================================================== */
/* ========================================================================== */
/*                                Revision History                            */
/* ========================================================================== */
/*    06/17/90    rlb        Initial Implementation                           */
/* ========================================================================== */

unsigned char menu_color = (BLUE << 4) | CYAN;
unsigned char warning_color = (CYAN << 4) | RED;
unsigned char error_color = (RED << 4) | WHITE;
unsigned char info_color = (CYAN << 4) | BLACK;
unsigned char keyhelp_color = (BLUE << 4) | CYAN;
unsigned char field_color = (WHITE << 4) | BLACK;
```

```
/* ========================================================================= */
/*                                 COLORS.H                                  */
/* ========================================================================= */
/*  This include file assigns the appropriate bit sequences to colors        */
/* ========================================================================= */
/*                            Created under Contract                         */
/*                                    to                                     */
/*                              EPA/AEERL/GECD/EMB                           */
/*                                                                           */
/*                                    by                                     */
/*                                                                           */
/*                              Radian Corporation                           */
/*                           Research Triangle Park, NC                      */
/* ========================================================================= */
/* ========================================================================= */
/*                              Revision History                             */
/* ========================================================================= */
/*    04/17/90   rlb       Initial Implementation                            */
/* ========================================================================= */

#define COLORS_DEFINED

#define BLACK     0
#define BLUE      1
#define GREEN     2
#define CYAN      3
#define RED       4
#define MAGENTA   5
#define BROWN     6
#define WHITE     7

#define INTENSE   8
#define BLINK     128

extern unsigned char menu_color ;
extern unsigned char warning_color ;
extern unsigned char error_color ;
extern unsigned char info_color ;
extern unsigned char keyhelp_color ;
extern unsigned char field_color;
```

```
/* ================================================================== */
/*                            CONSTANT.H                              */
/* ================================================================== */
/*    This include file defines math constants for use in calculation. */
/* ================================================================== */
/*                         Created under Contract                     */
/*                                  to                                */
/*                           EPA/AEERL/GECD/EMB                       */
/*                                                                    */
/*                                  by                                */
/*                                                                    */
/*                           Radian Corporation                       */
/*                        Research Triangle Park, NC                  */
/* ================================================================== */
/* ================================================================== */
/*                         Revision History                           */
/* ================================================================== */
/*    04/17/90   rlb       Initial Implementation                     */
/* ================================================================== */


#ifndef CONSTANTS
#define CONSTANTS

#define PI  3.14159265358979323846

/* ================================================================== */
/*              R - Universal Gas Constant  (J / (mole K))            */
/* ================================================================== */

#define R   8.3144

/* ================================================================== */
/*    Multiplication Factor to convert Cubic Ft to Cubic Meters       */
/* ================================================================== */

#define FT3_TO_M3      0.028316846

#endif
```

```
/* ========================================================================== */
/*                                 DEFAULTS.H                                 */
/* ========================================================================== */
/*    This include file defines all of the default values for the             */
/*    parameters used in the program.                                         */
/* ========================================================================== */
/*                           Created under Contract                           */
/*                                    to                                      */
/*                             EPA/AEERL/GECD/EMB                             */
/*                                                                            */
/*                                    by                                      */
/*                                                                            */
/*                             Radian Corporation                             */
/*                          Research Triangle Park, NC                        */
/* ========================================================================== */
/* ========================================================================== */
/*                             Revision History                               */
/* ========================================================================== */
/*    04/17/90    rlb        Initial Implementation                           */
/* ========================================================================== */


#ifndef DEFAULTS_DEFINED
#define DEFAULTS_DEFINED

#define DEFAULT_YEARS_PRIOR      10

#define DEFAULT_CH4              50.0
#define DEFAULT_CO2              50.0

#define DEFAULT_REGULATORY_L0    6000.0
#define DEFAULT_REGULATORY_K     0.05
#define DEFAULT_REGULATORY_NMOC  4000.0

#define DEFAULT_AP42_K           0.04
#define DEFAULT_AP42_L0          4411
#define DEFAULT_AP42_NMOC_CODIS  2420
#define DEFAULT_AP42_NMOC_NOCO   595

#define DEFAULT_WELL_DEPTH       0.0
#define DEFAULT_INFLUENCE_RAD    0.0
#define DEFAULT_REFUSE_DENSITY   0.018
#define DEFAULT_DECOMP_FRACTION  1.0
#define DEFAULT_FLOW_RATE        0.0
#define DEFAULT_REFUSE_AGE       0.0

#define DEFAULT_TEMP             293.0
#define DEFAULT_PRESSURE         1.01325E5

#define DEFAULT_NMOC_MOLE_WT     86.18    /*  Molecular Wt of NMOC is  */
                                          /*  established assuming it  */
                                          /*  is HEXANE.               */

#define YEARS_PASSED_CLOSURE     50
#define MAX_YEARS_PAST_CLOSURE 200
#define MAX_DURATION             300
#define MAX_AIR_TOXICS           60

#define LINES_PER_PAGE           66
#define PAGE_WIDTH               80

#endif
```

```
/* ========================================================================== */
/*                                  HELPASGN.H                                */
/* ========================================================================== */
/*    This include file assigns help numbers to all of the help tags.         */
/*      Help tags are all caps versions of the menu function names.           */
/* -------------------------------------------------------------------------- */
/*                            Created under Contract                          */
/*                                      to                                    */
/*                              EPA/AEERL/GECD/EMB                            */
/*                                                                            */
/*                                      by                                    */
/*                                                                            */
/*                              Radian Corporation                            */
/*                          Research Triangle Park, NC                        */
/* ========================================================================== */
/* ========================================================================== */
/*                              Revision History                              */
/* ========================================================================== */
/*    04/17/90   rlb       Initial Implementation                             */
/* ========================================================================== */

#ifndef HELP_DEFINED
#define HELP_DEFINED

#define MAIN_MENU              1
#define STUDY_TYPE             2
#define GET_STUDY_NAME         3
#define EDIT_MENU              4
#define EDIT_CHEM_COMP         5
#define AIR_TOXICS_MENU        6
#define K_SELECTION_MENU       7
#define EDIT_K_VALUE           8
#define CALCULATE_K_MENU       9
#define LO_SELECTION_MENU      10
#define EDIT_PERIODS           11
#define DISPLAY_RESULTS_MENU   12
#define TABULAR_REPORT_MENU    13
#define ON_SCREEN_GRAPHIC      14
#define REPORT_SELECTOR        15
#define SHOW_RESULTS           16
#define CONFIGURE_SYSTEM       17
#define DISPLAY_SAVE_MENU      18
#define GET_OUTPUT_FILENAME    19
#define DISPLAY_CLOSURE_YEAR   20
#define FILE_SELECTOR          21

extern char *mainmenu_help_string, *chemcomp_help_string,
            *airtoxmn_help_string, *generic_help_string,
            *generic_menu_help_string, *studname_help_string,
            *config_help_string, *continue_help_string,
            *tabular_report_key_help, *any_key_string,
            *abortable_menu_help_string, *abortable_warning_help_string,
            *no_default_help_string;

#endif
```

```
/* ========================================================================== */
/*                                LANDFILL.H                                  */
/* ========================================================================== */
/*    This include file defines the user interface information for the        */
/*    cscape library                                                          */
/* ========================================================================== */
/*                            Created under Contract                          */
/*                                     to                                     */
/*                              EPA/AEERL/GECD/EMB                            */
/*                                                                            */
/*                                     by                                     */
/*                                                                            */
/*                              Radian Corporation                            */
/*                           Research Triangle Park, NC                       */
/* ========================================================================== */
/* ========================================================================== */
/*                              Revision History                              */
/* ========================================================================== */
/*    04/17/90   rlb       Initial Implementation                             */
/* ========================================================================== */

#ifndef SPECIAL_FIELDS
#define SPECIAL_FIELDS

extern  void    susdbl_fenter(sed_type);
extern  boolean susdbl_fexit(sed_type);
extern  void    susdbl_fkey(sed_type);
extern  void    susdbl_senter(sed_type, int);
extern  void    susdbl_sexit(sed_type, int);

extern  struct  field_funcs susdbl_funcs;


extern  void    susint_fenter(sed_type);
extern  boolean susint_fexit(sed_type);
extern  void    susint_fkey(sed_type);
extern  void    susint_senter(sed_type, int);
extern  void    susint_sexit(sed_type, int);

extern  struct  field_funcs susint_funcs;


extern  void    dosstr_fenter(sed_type);
extern  boolean dosstr_fexit(sed_type);
extern  void    dosstr_fkey(sed_type);
extern  void    dosstr_senter(sed_type, int);
extern  void    dosstr_sexit(sed_type, int);

extern  struct  field_funcs dosstr_funcs;


extern  void    dosfile_fenter(sed_type);
extern  boolean dosfile_fexit(sed_type);
extern  void    dosfile_fkey(sed_type);
extern  void    dosfile_senter(sed_type, int);
extern  void    dosfile_sexit(sed_type, int);

extern  struct  field_funcs dosfile_funcs;


extern  void      myselect_fenter(sed_type);
extern  boolean myselect_fexit(sed_type);
extern  void      myselect_fkey(sed_type);
extern  void      myselect_senter(sed_type, int);

extern  struct  field_funcs myselect_funcs;
```



```
#endif
```

```
/* ========================================================================== */
/*                                LANDFILL.H                                  */
/* ========================================================================== */
/*    This include file defines the individual error messages displayed       */
/*    by the verification error screen.                                       */
/* ========================================================================== */
/*                           Created under Contract                           */
/*                                    to                                      */
/*                             EPA/AEERL/GECD/EMB                             */
/*                                                                            */
/*                                    by                                      */
/*                                                                            */
/*                             Radian Corporation                             */
/*                          Research Triangle Park, NC                        */
/* ========================================================================== */
/* ========================================================================== */
/*                             Revision History                               */
/* ========================================================================== */
/*    04/17/90   rlb       Initial Implementation                             */
/* ========================================================================== */

struct messages
               {
               char string1[40], string2[40], string3[40];
               };


struct messages error_msg[] = {
   {"Refuse in place is less", "than data in previous", "years"},
   {"Final value for refuse", "in place must be", "greater than zero"},
   {"Largest value cannot", "exceed permitted capacity", ""},
   {"Current year must be", "greater than the year", "opened"},
   {"Capacity cannot be zero", "", ""},
   {"Year of closure must be", "at least one year beyond", "the current year"},
   {"Operation cannot exceed", "internal limit of", "300 years"},
   {"Methane percentage", "cannot exceed", "100 percent"},
   {"Number of air toxics", "cannot exceed an internal", "limit of 20"},
   {"Methane percentage of", "landfill gas cannot","be zero"},
   {"Chemical name must be", "specified",""},
   {"Molecular weight must", "be greater than","zero"},
   {"Calculation must not", "exceed 200 years passed", "closure."},
   {"NMOC value cannot", "exceed 1,000,000 ppmv.", ""}
                              };
```

```
/* ========================================================================== */
/*                                  AIRTOXMN.C                                */
/* ========================================================================== */
/*  This file contains the code for displaying and altering the list of       */
/*  individual air toxics tracked by the program.                             */
/* ========================================================================== */
/*                            Created under Contract                          */
/*                                     to                                     */
/*                              EPA/AEERL/GECD/EMB                            */
/*                                                                            */
/*                                     by                                     */
/*                                                                            */
/*                              Radian Corporation                            */
/*                          Research Triangle Park, NC                        */
/* ========================================================================== */
/* ========================================================================== */
/*                              Revision History                              */
/* ========================================================================== */
/*    07/03/90    rlb       Initial Implementation                            */
/*    01/17/94    rlb       Altered to allow entry of a codisposal and        */
/*                             no codisposal emission factor.  Also,          */
/*                             specifies the printing format for double       */
/*                             values using the modification to fnsusdbl      */
/* ========================================================================== */

#include <stdio.h>
#include <cscape.h>
#include <scancode.h>
#include <string.h>
#include <ctype.h>
#include <malloc.h>
#include "landfill.h"
#include "landuser.h"
#include "helpasgn.h"
#include "defaults.h"
#include "colors.h"

/* ========================================================================== */
/*   Define window markers by the IBM ascii code representations              */
/* ========================================================================== */

#define UP_ARROW      24
#define DOWN_ARROW    25
#define SPACE         32

/* ========================================================================== */
/*                     This is the main menuing function.                     */
/*  Air_toxic_ptr is passed in as an pointer to an array pointer because      */
/*  this routine has a requirement to change the location of that list        */
/*  periodically as the list length changes and memory configurations         */
/*  change.                                                                   */
/* ========================================================================== */

int air_toxics_menu(air_toxic_ptr, number_air_toxics)
struct air_toxic_structure **air_toxic_ptr;
int *number_air_toxics;
{
extern struct system_parms config;
struct air_toxic_structure *air_toxic, *scratch_air_toxic;
menu_type   menu, choice_menu, key_menu;
sed_type    sed, choice_sed, key_sed;
int         ret, i, old_offset = 0, curr_offset,
            current_end, current_field, scratch, scratch_number;
char up_indicator = SPACE, down_indicator = SPACE;
char chemical[31];
char *ptr;
```



```
double mole_wt = 0.0;

int create_air_toxic_menu(menu_type *, sed_type *,
                          struct air_toxic_structure *);
int delete_air_toxics_menu(int, menu_type, sed_type);
int copy_air_toxic_list(struct air_toxic_structure **,
                        struct air_toxic_structure **, int);

extern void initialize_air_toxics(char *, char *,
                    struct air_toxic_structure **, int *);

   reset_insert_status();

/* ========================================================================== */
/*  Create a copy of the original air toxic list so we can change the         */
/*  information without destroying the original                               */
/* ========================================================================== */

   copy_air_toxic_list(&scratch_air_toxic, air_toxic_ptr,
                       *number_air_toxics);

   scratch_number = *number_air_toxics;


/* ========================================================================== */
/*                         Start defining the Cscape menu                     */
/* ========================================================================== */

   menu = menu_Open();

   build_scrollbox(menu, menu_color, 18, 78);

        menu_Printf(menu, "@p[1,24]Edit Air Pollutant Concentration");
        menu_Printf(menu, "@p[2,0]|@[76,-]|");
        menu_Printf(menu, "@p[3,0]|     ");
        menu_Printf(menu, "@p[3,35]Molecular");
        menu_Printf(menu, "@p[3,51]Concentration (ppm V)");
        menu_Printf(menu, "@p[4,0]|     ");
        menu_Printf(menu, "@p[4,14]Chemical");
        menu_Printf(menu, "@p[4,35] Weight    ");
        menu_Printf(menu, "@p[4,49]No Codisposal  Codisposal");
        menu_Printf(menu, "@p[5,0]|--@[74,-]|");

   menu_Printf(menu, "@p[16,1]@fp[#]", &up_indicator, &char_funcs);
   menu_Printf(menu, "@p[6,1]@fp[#]",  &down_indicator, &char_funcs);

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, menu_color, menu_color, menu_color);
   sed_SetPosition(sed, 3, 1);

   sed_Push(sed);

/* ========================================================================== */
/*  Before you draw the menu on the screen, check to see if the menu list     */
/*  is longer than the available window.  If so, set the down indicator       */
/*  to its on state and redraw the screen.                                    */
/* ========================================================================== */

   if (*number_air_toxics > 12) down_indicator = DOWN_ARROW;

   sed_Repaint(sed);
   show_key_help(&key_menu, &key_sed, airtoxmn_help_string);

   sed_SetActive(sed, TRUE);
```

```
/* ========================================================================= */
/*  Check to see if there is any air toxics in the list.  If so, create a */
/*  menu with them, draw it on the screen, and set it for entry.  This is */
/*  necessary because Cscape does not like menus with 0 entries.          */
/* ========================================================================= */

   if (scratch_number)
      {
      create_air_toxics_menu(&choice_menu, &choice_sed, scratch_air_toxic,
                             scratch_number);

      sed_Repaint(choice_sed);
      sed_SetActive(choice_sed, TRUE);
      sed_GotoFirstField(choice_sed);

      sed_DoFieldSenter(choice_sed, sed_GetFieldNo(choice_sed));
      sed_DoFieldFenter(choice_sed, sed_GetFieldNo(choice_sed));
      }

/* ========================================================================= */
/*                          This is the Main Loop                            */
/*     It will continue until the user presses FN10 or the ESC key.         */
/* ========================================================================= */

   do
      {

/* ========================================================================= */
/*  If there are any entries in the air toxic list, check to see if there */
/*  are too many for the menu screen, set the more indicators, and begin  */
/*  accepting field entry information.  If there are no entries, just get */
/*  the keys hit.                                                          */
/* ========================================================================= */

      if (scratch_number)
         {
         curr_offset = sed_GetYoffset(choice_sed);

         if ((curr_offset) && (scratch_number > 11))
            up_indicator = UP_ARROW;
         else
            up_indicator = SPACE;

         if (scratch_number - curr_offset > 11)
            down_indicator = DOWN_ARROW;
         else
            down_indicator = SPACE;

         if ((curr_offset != old_offset) || (ret == FN5) || (ret = FN6))
            {
            sed_RepaintFields(sed);
            old_offset = curr_offset;
            }

         ret = accept_menu_choices(choice_sed);
         }
      else
         ret = kb_Read();

/* ========================================================================= */
/*                          Evaluate key entered                             */
/* ========================================================================= */

      switch (ret)
         {
```



```
/* ========================================================================= */
/*  FN5 - Add a new air toxic.                                               */
/*                                                                           */
/*  First, check to see if they have maxed out on air toxics.  If not,       */
/*  call the routine to prompt for the information and check to see if       */
/*  they terminated with an F10 key (if not, stop).  If a list exists,       */
/*  delete it first.  Then create a new list and put the new info in it.     */
/*  Since we have got to have at least one entry now, recreate the menu,     */
/*  display it, and start accepting field entry from it.                     */
/* ========================================================================= */
         case FN5:
            if (scratch_number + 1 > MAX_AIR_TOXICS)
               {
               verify_error(8);
               break;
               }

            chemical[0] = '\0';
            mole_wt = 0.0;

            if (add_air_toxic(chemical, &mole_wt))
               {
               if (scratch_number)
                  delete_air_toxics_menu(scratch_number, choice_menu,
                                         choice_sed);

               scratch_air_toxic = (struct air_toxic_structure *)
                                   realloc(scratch_air_toxic,
                                     (scratch_number + 1) *
                                       sizeof(struct air_toxic_structure));

               strcpy(scratch_air_toxic[scratch_number].chemical_name, chemical)
;
               scratch_air_toxic[scratch_number].molecular_wt = mole_wt;
               scratch_air_toxic[scratch_number].concentration[NOCODISPOSAL] = 0
.0;
               scratch_air_toxic[scratch_number].concentration[CODISPOSAL] = 0.0
;

               create_air_toxics_menu(&choice_menu, &choice_sed,
                                      scratch_air_toxic, scratch_number + 1);

               sed_SetActive(choice_sed, TRUE);
               sed_GotoField(choice_sed, 4 * (scratch_number)++ + 2);

               sed_Repaint(choice_sed);

               sed_DoFieldSenter(choice_sed, sed_GetFieldNo(choice_sed));
               sed_DoFieldFenter(choice_sed, sed_GetFieldNo(choice_sed));
               }

            break;

/* ========================================================================= */
/*  FN6 - Delete an air toxic.                                               */
/*                                                                           */
/*  First, check to see if there is anything to delete. If so, get the       */
/*  current field number so we can determine which entry to delete.          */
/*  Next, delete the menu (which must exist since entries exist) and,        */
/*  if there is more than 1 entry in the list, call the function to loose   */
/*  the entry we are sitting on (since there are 3 fields for each entry     */
/*  and we are sitting on the 3rd of 3 fields, the entry must be the          */
/*  current field divided by 3), otherwise, loose the entire list.  Now,     */
/*  check to make sure there is still entries in the list now and, if so,    */
/*  build the menu, draw it, and get field input.                            */
/* ========================================================================= */
```

```
         case FN6:
            if (scratch_number)
               {
               current_field = sed_GetFieldNo(choice_sed);
               delete_air_toxics_menu(scratch_number, choice_menu,
                                      choice_sed);

               if (scratch_number > 1)
                  delete_airtox_entry(&scratch_air_toxic, current_field / 4,
                                      scratch_number);
               else
                  {
                  free(scratch_air_toxic);
                  scratch_air_toxic = NULL;
                  }

               (scratch_number)--;
               }

            if (scratch_number)
               {
               create_air_toxics_menu(&choice_menu, &choice_sed,
                                      scratch_air_toxic, scratch_number);
               sed_SetActive(choice_sed, TRUE);
               scratch = current_field - 4;
               sed_GotoField(choice_sed, (scratch <= 1) ? 2 : scratch);
               sed_Repaint(choice_sed);
               }

            break;

/* ========================================================================= */
/*  FN2 - Reset the Air Toxic List to Defaults                               */
/*                                                                           */
/*  Pop up a warning screen.  If they tell you to go ahead, delete the       */
/*  menu (if there is one) and free the memory.  Reinitialize the list       */
/*  by re-reading the air toxics file and build the new menu.                */
/* ========================================================================= */
         case FN2:
            if (warn_about_reset())
               {
               if (scratch_number)
                  delete_air_toxics_menu(scratch_number, choice_menu,
                                         choice_sed);

               if (scratch_air_toxic != NULL) free(scratch_air_toxic);
               initialize_air_toxics(config.landfill_dir, "airtoxic.txt",
                                     &scratch_air_toxic, &scratch_number);

               if (scratch_number)
                  {
                  create_air_toxics_menu(&choice_menu, &choice_sed,
                                         scratch_air_toxic, scratch_number);
                  sed_SetActive(choice_sed, TRUE);
                  sed_GotoFirstField(choice_sed);
                  sed_Repaint(choice_sed);
                  }
               }

            break;

/* ========================================================================= */
/*  FN1 - Show the Help Information for this screen                          */
/* ========================================================================= */

         case FN1:
```



```
            compile_help(AIR_TOXICS_MENU);
            break;
         }
      }
   while ((ret != FN10) && (ret) && (ret != ESC));

/* ========================================================================= */
/*                                 End of Loop.                              */
/*  Note: there are 3 conditions being tested.  This is required because     */
/*  two different routines are being called as necessary to get input        */
/*  from the menu.  sed_Go and kb_Read both return FN10.  However, sed_Go    */
/*  returns 0 when ESC is return while kb_Read returns ESC.                  */
/* ========================================================================= */

/* ========================================================================= */
/*  If the loop was exited with entries in the list, make sure the fields    */
/*  have all been converted and delete the menu.                             */
/* ========================================================================= */
   if (scratch_number)
      {
      sed_DoFieldSexit(choice_sed, sed_GetFieldNo(choice_sed));

      sed_SetActive(choice_sed, FALSE);
      delete_air_toxics_menu(scratch_number, choice_menu, choice_sed);
      }

/* ========================================================================= */
/*  The loop was exited because FN10 was pressed.  That means the data       */
/*  was accepted and we should copy the scratch data areas we created        */
/*  into the real ones.                                                      */
/* ========================================================================= */

   if (ret == FN10)
      {
      *number_air_toxics = scratch_number;

/* ========================================================================= */
/*  Free up the old list's memory and create a new list with the new         */
/*  number of entries.                                                       */
/* ========================================================================= */

      if (*air_toxic_ptr != NULL)
         {
         free(*air_toxic_ptr);
         *air_toxic_ptr = NULL;
         }

      copy_air_toxic_list(air_toxic_ptr, &scratch_air_toxic,
                          *number_air_toxics);
      }

/* ========================================================================= */
/*                  Deallocate the scratch lists memory.                     */
/* ========================================================================= */

   if (scratch_number) free(scratch_air_toxic);

/* ========================================================================= */
/*        Delete the menu and return with the key they exited on.            */
/* ========================================================================= */
   sed_SetActive(sed, FALSE);

   destroy_key_help(key_menu, key_sed);
   sed_Pop(sed);

   menu_Close(menu);
```

```
   sed_Close(sed);

   return(ret);
}



/* ========================================================================== */
/*  This function makes space for and copies an original air toxics list  */
/*  to a new list.                                                          */
/* ========================================================================== */

int copy_air_toxic_list(dest_ptr, source_ptr, count)
struct air_toxic_structure **dest_ptr, **source_ptr;
int count;
{
struct air_toxic_structure *dest_list, *source_list;
int i;

   source_list = *source_ptr;

   if (count == 0)
      {
      *dest_ptr = NULL;
      return(0);
      }

   dest_list = *dest_ptr = (struct air_toxic_structure *)
               calloc(count, sizeof(struct air_toxic_structure));

   if (dest_list == NULL)
      show_mem_error("copy_air_toxic_list", "dest_ptr");

   for (i = 0 ; i < count ; i++)
      {
      strcpy(dest_list[i].chemical_name, source_list[i].chemical_name);
      dest_list[i].molecular_wt = source_list[i].molecular_wt;
      dest_list[i].concentration[NOCODISPOSAL]
            = source_list[i].concentration[NOCODISPOSAL];
      dest_list[i].concentration[CODISPOSAL]
            = source_list[i].concentration[CODISPOSAL];
      }

   return(count);
}


/* ========================================================================== */
/*  This function builds the Cscape type menu from the air toxics list.   */
/*  It also sizes it so it can fit in the space allowed by the other sed. */
/* ========================================================================== */

int create_air_toxics_menu(choice_menu, choice_sed, air_toxic,
                           number_air_toxics)
menu_type *choice_menu;
sed_type *choice_sed;
struct air_toxic_structure *air_toxic;
int number_air_toxics;
{
int i;
char format[80];

   *choice_menu = menu_Open();

   for (i = 0 ; i < number_air_toxics ; i++)
      {
```



```
      menu_Printf(*choice_menu, "@p[%d,0]@c[%c]@[72, ]", i, menu_color);

      sprintf(format, "@p[%d,0]@c[%c]@fp[@[30,#]]", i, menu_color);
      menu_Printf(*choice_menu, format,
                  air_toxic[i].chemical_name, &string_funcs);

      sprintf(format, "@p[%d,33]@c[%c]@fpd3[@[7,#]]", i, menu_color);
      menu_Printf(*choice_menu, format,
                   &(air_toxic[i].molecular_wt), &susdbl_funcs,
                   NULL, NULL, "%6.2lf");

      sprintf(format, "@p[%d,47]@c[%c]@fd3[@[10,#]]", i, menu_color);
      menu_Printf(*choice_menu, format,
                  &(air_toxic[i].concentration[NOCODISPOSAL]),
                  &susdbl_funcs, NULL, NULL, "%.4lf");

      sprintf(format, "@p[%d,62]@c[%c]@fd3[@[10,#]]", i, menu_color);
      menu_Printf(*choice_menu, format,
                  &(air_toxic[i].concentration[CODISPOSAL]),
                  &susdbl_funcs, NULL, NULL, "%.4lf");
      }

   menu_Flush(*choice_menu);

   *choice_sed = sed_Open(*choice_menu);

   sed_SetHeight(*choice_sed, 11);
   sed_SetWidth(*choice_sed, 72);

   sed_SetColors(*choice_sed, menu_color, menu_color, field_color);

   sed_SetPosition(*choice_sed, 9, 4);

   sed_Push(*choice_sed);

   sed_UpdateFields(*choice_sed);
}



/* ========================================================================== */
/*                This function deletes a Cscape menu.                        */
/* ========================================================================== */

int delete_air_toxics_menu(count, choice_menu, choice_sed)
int count;
menu_type choice_menu;
sed_type choice_sed;
{
   sed_Pop(choice_sed);

   menu_Close(choice_menu);
   sed_Close(choice_sed);
}


/* ========================================================================== */
/*  This function deletes a given entry (denoted by index number) from    */
/*  the air toxics list.                                                    */
/* ========================================================================== */

int delete_airtox_entry(air_toxics_ptr, entry, number_of_entries)
struct air_toxic_structure **air_toxics_ptr;
int entry, number_of_entries;
{
struct air_toxic_structure *air_toxics, *new_air_toxics;
```

```
int i, j;

/* ========================================================================== */
/*  Make a copy the the original pointer.  Then, make the original            */
/*  point to a new area of memory allocated as one air toxic entry            */
/*  smaller than before.                                                      */
/* ========================================================================== */
   air_toxics = *air_toxics_ptr;

   *air_toxics_ptr = (struct air_toxic_structure *)
      calloc((size_t) (number_of_entries - 1),
            sizeof(struct air_toxic_structure));

   if (*air_toxics_ptr == NULL)
      show_mem_error("delete_airtox_entry", "air_toxics_ptr");

/* ========================================================================== */
/*  Copy the original list to the new list, skipping the desired entry.       */
/* ========================================================================== */

   new_air_toxics = *air_toxics_ptr;

   for (j = 0, i = 0 ; i < number_of_entries ; i++)
      if (i != entry)
         {
         strcpy(new_air_toxics[j].chemical_name, air_toxics[i].chemical_name);
         new_air_toxics[j].molecular_wt = air_toxics[i].molecular_wt;
         new_air_toxics[j].concentration[NOCODISPOSAL]
               = air_toxics[i].concentration[NOCODISPOSAL];
         new_air_toxics[j].concentration[CODISPOSAL]
               = air_toxics[i].concentration[CODISPOSAL];
         j++;
         }

/* ========================================================================== */
/*                            Free the old list                               */
/* ========================================================================== */

   free(air_toxics);

   return(number_of_entries - 1);
}


/* ========================================================================== */
/*  This function handles all the Cscape key trapping and returns the         */
/*  character received.                                                       */
/* ========================================================================== */

int accept_menu_choices(choice_sed)
sed_type choice_sed;
{
int ret;

   sed_DoFieldFkey(choice_sed, sed_GetFieldNo(choice_sed));

   ret = sed_GetBaton(choice_sed);

   sed_SetBaton(choice_sed, -1);

   return(ret);
}


/* ========================================================================== */
/*  This function activates a pop-up to receive the information on the        */
```



```
/*  new air toxic being added to the list.                                    */
/* ========================================================================== */

int add_air_toxic(chemical, mole_wt)
char *chemical;
double *mole_wt;
{
menu_type   menu;
sed_type    sed;
int         ret;

/* ========================================================================== */
/*                    Zero the string before editing.                         */
/* ========================================================================== */

   chemical[0] = '\0';

/* ========================================================================== */
/*                          Define the Cscape Menu.                           */
/* ========================================================================== */

   menu = menu_Open();

   menu_Printf(menu, "@p[0,0]@c{%c}+@[43,-)+", menu_color);
   menu_Printf(menu, "@p[1,0]¦");
   menu_Printf(menu, "@p[1,14]Add an Air Toxic");
   menu_Printf(menu, "@p[1,44]¦");
   menu_Printf(menu, "@p[2,0]¦@[43,-]¦");
   menu_Printf(menu, "@p[3,0]¦");
   menu_Printf(menu, "@p[3,44]¦");
   menu_Printf(menu, "@p[4,0]¦ Air Toxic :");
   menu_Printf(menu, "@p[4,44]¦");
   menu_Printf(menu, "@p[5,0]¦");
   menu_Printf(menu, "@p[5,44]¦");
   menu_Printf(menu, "@p[6,0]¦ Molecular Weight : ");
   menu_Printf(menu, "@p[6,44]¦");
   menu_Printf(menu, "@p[7,0]¦");
   menu_Printf(menu, "@p[7,44]¦");
   menu_Printf(menu, "@p[8,0]+@[43,-]+");
   menu_Printf(menu, "@p[4,14]@f[@[29,#]]", chemical, &string_funcs);
   menu_Printf(menu, "@p[6,21]@f[@[16,#]]", mole_wt, &susdbl_funcs);

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, menu_color, menu_color, field_color);
   sed_SetPosition(sed, 7, 17);

   sed_Push(sed);
   sed_Repaint(sed);

/* ========================================================================== */
/*        This is the main loop.  It terminates on ESC or F10.                */
/* ========================================================================== */

   do
      {
      ret = sed_Go(sed);

      if (ret) ret = verify_airtox_entry(chemical, mole_wt);
      }
   while ((ret) && (ret != FN10));

   sed_Pop(sed);

   menu_Close(menu);
```

```
   sed_Close(sed);
   return(ret);
}



int verify_airtox_entry(name, wt)
char *name;
double *wt;
{
   if (! strlen(name))
      {
      verify_error(10);
      return(1);
      }

   if (*wt <= 0.0)
      {
      verify_error(11);
      return(1);
      }

   return(FN10);
}
```

```
/* ========================================================================== */
/*                                BUILDBOX.C                                  */
/* ========================================================================== */
/*       This file contains the code for building various box types           */
/* ========================================================================== */
/*                             Created under Contract                         */
/*                                       to                                   */
/*                               EPA/AEERL/GECD/EMB                           */
/*                                                                            */
/*                                       by                                   */
/*                                                                            */
/*                               Radian Corporation                           */
/*                            Research Triangle Park, NC                      */
/* ========================================================================== */
/* ========================================================================== */
/*                               Revision History                             */
/* ========================================================================== */
/*    07/03/90   rlb        Initial Implementation                            */
/* ========================================================================== */

#include <stdio.h>
#include <cscape.h>

int buildbox(menu, color, lines, width)
menu_type menu;
int lines, width;
unsigned char color;
{
int i;

   menu_Printf(menu, "@p[0,0]@c[%c]+@[%d,-]+", color, width - 2);
   menu_Printf(menu, "@p[1,0]¦@[%d, ]¦", width - 2);
   menu_Printf(menu, "@p[2,0]¦@[%d,-]¦", width - 2);

   for (i = 3 ; i <= lines - 2 ; i++)
      menu_Printf(menu, "@p[%d,0]¦@[%d, ]¦", i, width - 2);

   menu_Printf(menu, "@p[%d,0]+@[%d,-]+", lines - 1, width - 2);

   return(0);
}


int build_scrollbox(menu, color, lines, width)
menu_type menu;
int lines, width;
unsigned char color;
{
int i;

   menu_Printf(menu, "@p[0,0]@c[%c]+@[%d,-]+", color, width - 2);
   menu_Printf(menu, "@p[1,0]¦@[%d, ]¦", width - 2);
   menu_Printf(menu, "@p[2,0]¦--@[%d,-]¦", width - 4);

   for (i = 3 ; i <= lines - 2 ; i++)
      menu_Printf(menu, "@p[%d,0]¦ ¦@[%d, ]¦", i, width - 4);

   menu_Printf(menu, "@p[%d,0]+--@[%d,-]+", lines - 1, width - 4);

   return(0);
}
```

```
/* ================================================================== */
/*                            CALCMAIN.C                              */
/* ================================================================== */
/*  This file contains the routines used to calculate the landfill gas */
/*  emissions.                                                        */
/* ================================================================== */
/*                         Created under Contract                     */
/*                                 to                                 */
/*                           EPA/AEERL/GECD/EMB                       */
/*                                                                    */
/*                                 by                                 */
/*                                                                    */
/*                           Radian Corporation                       */
/*                       Research Triangle Park, NC                   */
/* ================================================================== */
/* ================================================================== */
/*                           Revision History                         */
/* ================================================================== */
/*    07/03/90   rlb      Initial Implementation                      */
/*    01/21/94   rlb      Changed routine to use include_air_toxics flag */
/*                           to select codisposal option.             */
/* ================================================================== */

#include <stdio.h>
#include <math.h>
#include <malloc.h>
#include <cscape.h>
#include <scancode.h>
#include "landfill.h"
#include "landuser.h"
#include "defaults.h"
#include "constant.h"
#include "colors.h"

/* ================================================================== */
/*              This is the main calculation function.                */
/*  This function takes care of all of the results memory management as */
/*  well as coordinating the calculation of the emission information.  */
/* ================================================================== */

int calculation_main(landfill_info, model_info, results, done)
struct landfill_parms *landfill_info;
struct model_parms *model_info;
struct model_results *results;
boolean *done;
{
menu_type status_menu, running_menu, key_menu;
sed_type status_sed, running_sed, key_sed;
int i, index, j, year;
double ch4_rate;
double model_function();

double mass_to_volume();
double volume_to_mass();

/* ================================================================== */
/*  Start out by setting the flag indicating whether the calculation was */
/*  done or not to FALSE.  This will not get reset until the calculation */
/*  is over.                                                          */
/* ================================================================== */

   *done = FALSE;

/* ================================================================== */
/*  Build the status box.  This box will hold notices for the user on the */
/*  action currently being taken by the computer.  Initialize the year */
```



```
/*  variable at the first year being calculated.  This will be changed */
/*  and displayed to let the user know how far the calculation has    */
/*  progressed                                                        */
/* ================================================================== */

   build_status(&status_sed, &status_menu);

   year = landfill_info->year_opened + 1;

/* ================================================================== */
/*  This is the major memory management section.  First, clear any data */
/*  left over from a previous run.  Then, make new space the length of */
/*  new calculation for each of the tracked variables.                */
/*                                                                    */
/*  There is an inherent advantage to callocing array space a bit at a */
/*  time - you can asked for memory spaning more than one segment total. */
/* ================================================================== */

   clear_previous_results(results);

   results->year_count = landfill_info->year_closed -
         landfill_info->year_opened + config.years_passed_closure;

   results->air_toxics_count = model_info->number_air_toxics;

   results->year_list = (int *) calloc((size_t) results->year_count,
                                       sizeof(int));

   if (results->year_list == NULL)
      {
      clear_previous_results(results);
      show_mem_error2();
      destroy_status(&status_sed, &status_menu);
      return(-1);
      }

   results->annual_ch4 = (double *) calloc((size_t) results->year_count,
      sizeof(double));
   if (results->annual_ch4 == NULL)
      {
      clear_previous_results(results);
      show_mem_error2();
      destroy_status(&status_sed, &status_menu);
      return(-1);
      }

   results->annual_co2 = (double *) calloc((size_t) results->year_count,
                                           sizeof(double));
   if (results->annual_co2 == NULL)
      {
      clear_previous_results(results);
      show_mem_error2();
      destroy_status(&status_sed, &status_menu);
      return(-1);
      }

   results->annual_nmoc = (double *) calloc((size_t) results->year_count,
                                            sizeof(double));
   if (results->annual_nmoc == NULL)
      {
      clear_previous_results(results);
      show_mem_error2();
      destroy_status(&status_sed, &status_menu);
      return(-1);
      }
```

```
/* ================================================================== */
/*  Only make space for air toxics if they are included in the calc.  */
/*                                                                    */
/*  The arrangement of memory for the air toxic emission is an array  */
/*  of pointers for each year of the calculation pointing to an array of */
/*  doubles containing the emissions for each air toxic calculated.   */
/* ================================================================== */

   if (model_info->include_air_toxics != NO_AT)
      {
      results->annual_air_toxics = (double **)
         calloc((size_t) model_info->number_air_toxics, sizeof(double *));
      if (results->annual_air_toxics == NULL)
         {
         clear_previous_results(results);
         show_mem_error2();
         destroy_status(&status_sed, &status_menu);
         return(-1);
         }

      for (i = 0 ; i < model_info->number_air_toxics ; i++)
         {
         results->annual_air_toxics[i] = (double *)
            calloc((size_t) results->year_count, sizeof(double));

         if (results->annual_air_toxics[i] == NULL)
            {
            clear_previous_results(results);
            show_mem_error2();
            destroy_status(&status_sed, &status_menu);
            return(-1);
            }
         }
      }

/* ================================================================== */
/*  Allocate space for the refuse acceptance history of the landfill. */
/*  This array holds acceptance rate for each year until the last year */
/*  the calculation is done.                                          */
/* ================================================================== */

   results->refuse_data = (double *) calloc((size_t) results->year_count,
                  sizeof(double));

   if (results->refuse_data == NULL)
      {
      clear_previous_results(results);
      show_mem_error2();
      destroy_status(&status_sed, &status_menu);
      return(-1);
      }

/* ================================================================== */
/*  Fill the acceptance rate list with the past history information and */
/*  project forward until the last calculation year.                  */
/* ================================================================== */

   get_refuse_list(landfill_info, results->refuse_data);

/* ================================================================== */
/*  If we have gotten this far, the memory allocation steps worked and we */
/*  are ready to do the calculation.  Pop-up a box to tell the user which */
/*  year we are currently calculating for.                            */
/* ================================================================== */

   build_running(&running_sed, &running_menu, &year);
```



```
   show_key_help(&key_menu, &key_sed, "ESC = Interrupt");

/* ================================================================== */
/*  Set a flag that tells if the calculation was completed without    */
/*  user interruption.                                                */
/* ================================================================== */

   *done = TRUE;

/* ================================================================== */
/*                            Main Loop                               */
/*  This loop calculates the methane, carbon dioxide, nmoc, and air toxic */
/*  emissions (if any) from the year after landfill start until the last */
/*  desired calculation year (10 years past closure).                 */
/* ================================================================== */

   for (i = 0 ; i < results->year_count ; i++, year++)
      {
      results->year_list[i] = year;
/* ================================================================== */
/*       Update the status box with the current calculation year      */
/* ================================================================== */

      sed_RepaintFields(running_sed);

/* ================================================================== */
/*            Get the current methane generation rate (Cft/yr)        */
/* ================================================================== */
      ch4_rate = model_function(model_info->k, model_info->l0,
                                results->refuse_data, i);

      results->annual_ch4[i] = ch4_rate;

/* ================================================================== */
/*  The carbon dioxide generation rate is assumed to be the balance of */
/*  landfill gas after accounting for the methane (Cft/yr).           */
/* ================================================================== */

      results->annual_co2[i] = ch4_rate * model_info->co2
                               / model_info->ch4;

/* ================================================================== */
/*  NMOC (non-methane organic compounds) are assumed to be a negligible */
/*  constituent of landfill gas.  The emission is expected to be directly */
/*  proportional to the landfill gas generation rate (the 1E6 compensates */
/*  for the NMOC entry units of ppmv.  Units in Cft/yr.               */
/* ================================================================== */

      results->annual_nmoc[i] = ch4_rate * 100.0 * model_info->nmoc
                                    / (1.0E6 * model_info->ch4);

/* ================================================================== */
/*                  Convert saved results into Mg/yr                  */
/* ================================================================== */

      results->annual_nmoc[i] =
         volume_to_mass(results->annual_nmoc[i] * FT3_TO_M3,
                        DEFAULT_NMOC_MOLE_WT, DEFAULT_TEMP,
                        DEFAULT_PRESSURE) / 1000.0;

/* ================================================================== */
/*  If air toxic analysis is selected, calculate the emission using the */
/*  the same assumption as in the NMOC case.                          */
/* ================================================================== */

      if (model_info->include_air_toxics != NO_AT)
```

```
         for (j = 0 ; j < model_info->number_air_toxics ; j++)
            results->annual_air_toxics[j][i] =
                ch4_rate * 100.0
                  * model_info->air_toxics[j].concentration[model_info->include_
air_toxics] /
                    (1.0E6 * model_info->ch4);

/* ================================================================== */
/*  Watch the keyboard.  If the user presses ESC, he wants to abort the  */
/*  calculation.  Make sure to set the done flag to FALSE to prevent     */
/*  displaying of the incomplete calculation set.                        */
/* ================================================================== */

      if (kb_Check())
         if (kb_Read() == ESC)
            {
            *done = FALSE;
            break;
            }
      }

/* ================================================================== */
/*                              End Loop                              */
/* ================================================================== */

/* ================================================================== */
/*                        Get rid of the status box                   */
/* ================================================================== */

   destroy_key_help(key_menu, key_sed);
   destroy_running(&running_sed, &running_menu);
   destroy_status(&status_sed, &status_menu);

/* ================================================================== */
/*  If the calculation was done completely, analyze the data and produce  */
/*  an executive summary of the results on the screen.                    */
/* ================================================================== */

   if (*done) display_summary(landfill_info, results);
}



/* ================================================================== */
/*  This function performs the dirty work of cleaning up the memory left  */
/*  over from a previous model run.                                       */
/* ================================================================== */

int clear_previous_results(results)
struct model_results *results;
{
int i;

   if (results->year_list != NULL) free(results->year_list);
   if (results->refuse_data != NULL) free(results->refuse_data);

   if (results->annual_ch4 != NULL) free(results->annual_ch4);
   if (results->annual_co2 != NULL) free(results->annual_co2);
   if (results->annual_nmoc != NULL) free(results->annual_nmoc);

/* ================================================================== */
/*  Cleaning up the air toxics array requires individually results for    */
/*  each one of the air toxics in the study.                              */
/* ================================================================== */
```



```
   for (i = 0 ; i < results->air_toxics_count ; i++)
      if (results->annual_air_toxics[i] != NULL)
         free(results->annual_air_toxics[i]);

   if (results->annual_air_toxics != NULL) free(results->annual_air_toxics);

   results->year_count = 0;
   results->air_toxics_count = 0;

   results->year_list = NULL;
   results->refuse_data = NULL;

   results->annual_ch4 = NULL;
   results->annual_co2 = NULL;
   results->annual_nmoc = NULL;
   results->annual_air_toxics = NULL;

   return(0);
}


/* ================================================================== */
/*  These functions use the ideal gas law to convert mass to volume and   */
/*                        volume to mass.                                 */
/*      All parameters are assumed to be in the correct units (MKS).      */
/* ================================================================== */


double mass_to_volume(mass, mole_wt, temp, pressure)
double mass, mole_wt, temp, pressure;
{
   return(mass * (1000.0 * R) * temp / (mole_wt * pressure));
}



double volume_to_mass(volume, mole_wt, temp, pressure)
double volume, mole_wt, temp, pressure;
{
   return((mole_wt * pressure * volume) / (1000.0 * R * temp));
}


/* ================================================================== */
/*  This is the actual model function.   It attempts to model the methane */
/*  emissions by assuming the generation is being emitted exponentially   */
/*  at different positions allow the emission curve by each subset of     */
/*  waste materials, depending on the time between the current year and   */
/*  the initial placement of the waste subset.  The emissions for each    */
/*  submass is calculated and summed to determine the total emission of   */
/*  the landfill as a whole.                                              */
/*                                                                        */
/*  Table of Variables                                                    */
/*                                                                        */
/*  k       The methane generation decay rate constant.                   */
/*  L0      The methane generation capacity constant.                     */
/*  refuse  The refuse acceptance rate list starting from the opening     */
/*          to the end calc year.                                         */
/*  i       The current year index (0 = 1 year after opening)             */
/* ================================================================== */

double model_function(k, l0, refuse, i)
double l0, k, *refuse;
int i;
{
int j;
```

```
double sum = 0;

   for (j = 0 ; j <= i ; j++)
      sum += refuse[j] * exp(-k * (i - j));

   return(k * 10 * sum);
}


/* ================================================================== */
/*  This function pieces together the landfill refuse acceptance rate */
/*  list through the last calculation day using the user entered refuse */
/*  in place list and the closure data.                               */
/* ================================================================== */

int get_refuse_list(data, refuse)
struct landfill_parms *data;
double *refuse;
{
int i, j, index_closed;
double sum = 0;

   index_closed = data->year_closed - data->year_opened - 1;

   sum += refuse[0] = data->refuse_in_place[0];

/* ================================================================== */
/*  Starting with the refuse in place list, get the difference between */
/*  successive years and place that in the acceptance rate list.  Keep */
/*  doing this until we reach the current year or the closure year.   */
/* ================================================================== */

   for (i = 1 ; (i < data->number_of_periods) && (i < index_closed) ; i++)
      {
      refuse[i] = data->refuse_in_place[i] - data->refuse_in_place[i - 1];
      sum += refuse[i];
      }

/* ================================================================== */
/*  For the balance of the list, fill it in with the last rate necessary */
/*  to close by closure year (determine in the periods entry process). */
/* ================================================================== */

   for (j = 0; j < (data->year_closed - data->current_year - 1) ; j++, i++)
      {
      refuse[i] = data->last_rate;
      sum += refuse[i];
      }

/* ================================================================== */
/*  Put the balance of the refuse required to reach capacity in the   */
/*  closure cell.  (This step was done because the original method of */
/*  calculating the acceptance rate break down was based on the last  */
/*  observed acceptance rate continuing until closure.  Consequently, the */
/*  balance of the list remaining just before closure may be smaller than */
/*  the actual last rate observed and needed to be calculated.  By    */
/*  leaving this step, it will compensate for changes in the closure  */
/*  algorithm.)                                                       */
/* ================================================================== */

   if (index_closed > 1)
      refuse[i++] = data->capacity - sum;

/* ================================================================== */
/*  Fill in the remaining part of the list with zeros (no waste accepted) */
/* ================================================================== */
```



```
   for (j = 0 ; j < config.years_passed_closure ; j++, i++)
      refuse[i] = 0.0;
}


/* ================================================================== */
/*  This function actually builds the status box.  It originally contains */
/*  a memory allocation message.  This gets overwritten by the year   */
/*  later.                                                            */
/* ================================================================== */

int build_status(sed, menu)
menu_type   *menu;
sed_type    *sed;
{

   *menu = menu_Open();

   menu_Printf(*menu, "@p[0,0]@c[%c]+@[36,-]+", info_color);
   menu_Printf(*menu, "@p[1,0]¦");
   menu_Printf(*menu, "@p[1,8]Calculation in Progress");
   menu_Printf(*menu, "@p[1,37]¦");
   menu_Printf(*menu, "@p[2,0]¦@[36,-]¦");
   menu_Printf(*menu, "@p[3,0]¦");
   menu_Printf(*menu, "@p[3,37]¦");
   menu_Printf(*menu, "@p[4,0]¦");
   menu_Printf(*menu, "@p[4,9]Allocating Memory");
   menu_Printf(*menu, "@p[4,37]¦");
   menu_Printf(*menu, "@p[5,0]¦");
   menu_Printf(*menu, "@p[5,37]¦");
   menu_Printf(*menu, "@p[6,0]¦");
   menu_Printf(*menu, "@p[6,10]Please Wait ...");
   menu_Printf(*menu, "@p[6,37]¦");
   menu_Printf(*menu, "@p[7,0]¦");
   menu_Printf(*menu, "@p[7,37]¦");
   menu_Printf(*menu, "@p[8,0]+@[36,-]+");

   menu_Flush(*menu);

   *sed = sed_Open(*menu);
   sed_SetColors(*sed, info_color, info_color, info_color);
   sed_SetPosition(*sed, 7, 20);

   sed_Push(*sed);
   sed_Repaint(*sed);

   return(0);
}


/* ================================================================== */
/*  This function builds the year status display information.  It is  */
/*  overlayed on the status box.                                      */
/* ================================================================== */

int build_running(sed, menu, year)
menu_type   *menu;
sed_type    *sed;
int *year;
{
int i;

   *menu = menu_Open();
```

```
   for (i = 0 ; i < 5 ; i++)
      menu_Printf(*menu, "@p[%d,0]@c[%c]@[35, ]", i, info_color);

   menu_Printf(*menu, "@p[1,12]@c[%c]Current Year", info_color);
   menu_Printf(*menu, "@p[3,16]@f[####]",
     year, &susint_funcs);

   menu_Flush(*menu);

   *sed = sed_Open(*menu);
   sed_SetColors(*sed, info_color, info_color, info_color);
   sed_SetPosition(*sed, 10, 21);

   sed_SetWidth(*sed, 35);
   sed_SetHeight(*sed, 5);

   sed_Repaint(*sed);

   return(0);
}

/* ========================================================================= */
/*   The following two functions destroy the menu items created in the     */
/*   above functions.                                                        */
/* ========================================================================= */


int destroy_running(sed, menu)
sed_type *sed;
menu_type *menu;
{
   menu_Close(*menu);
   sed_Close(*sed);
   return(0);
}


int destroy_status(sed, menu)
sed_type *sed;
menu_type *menu;
{
   sed_Pop(*sed);

   menu_Close(*menu);
   sed_Close(*sed);
   return(0);
}
```

```
/* ================================================================== */
/*                              CHEMCOMP.C                            */
/* ================================================================== */
/*  This file contains the code for displaying and altering the chemical  */
/*  composition of landfill gas (not including individual air toxics).    */
/* ================================================================== */
/*                         Created under Contract                     */
/*                                  to                                */
/*                           EPA/AEERL/GECD/EMB                       */
/*                                                                    */
/*                                  by                                */
/*                                                                    */
/*                           Radian Corporation                       */
/*                        Research Triangle Park, NC                  */
/* ================================================================== */
/* ================================================================== */
/*                          Revision History                          */
/* ================================================================== */
/*    07/03/90   rlb       Initial Implementation                     */
/*    02/08/94   rlb       Modified to permit default NMOC selections.*/
/*    04/14/94   rlb       Modified to change screen order to better  */
/*                         support the NMOC selection.  Added new     */
/*                         warning screen.                            */
/* ================================================================== */

#include <stdio.h>
#include <cscape.h>
#include <scancode.h>
#include <string.h>
#include "landfill.h"
#include "defaults.h"
#include "landuser.h"
#include "helpasgn.h"
#include "colors.h"

#define YES  1
#define NO   0

/* ================================================================== */
/*              Declare the choices for this selection                */
/* ================================================================== */

static char *include_ptrs[] = {"No Codisposal", "Codisposal", "None",
                               NULL};


/* ================================================================== */
/*     This is the main menuing function for edit chemical composition    */
/* ================================================================== */

int edit_chem_comp(model_info)
struct model_parms *model_info;
{
void (*old_fkey)();
struct model_parms scratch_model_parms;
menu_type   menu, key_menu;
sed_type    sed, key_sed;
int         ret, old_include_at;
struct select_struct air_toxic_choice;

int copy_model_parms(struct model_parms *, struct model_parms *);
void recalc_fkey();

   reset_insert_status();

   old_fkey = susdbl_funcs.fkey;
```



```
   susdbl_funcs.fkey = recalc_fkey;

/* ================================================================== */
/*        Copy the original model parameters into a scratch area.     */
/* ================================================================== */

   copy_model_parms(&scratch_model_parms, model_info);

/* ================================================================== */
/*       Define the contents of the Cscape toggle field function.     */
/* ================================================================== */

   old_include_at = air_toxic_choice.number = scratch_model_parms.include_air_to
xics;
   air_toxic_choice.list = include_ptrs;

/* ================================================================== */
/*                          Define the Cscape Menu                    */
/* ================================================================== */

   menu = menu_Open();

   buildbox(menu, menu_color, 15, 48);
   menu_Printf(menu, "@p[1,11]Edit Chemical Composition");
   menu_Printf(menu, "@p[4,5]Air Pollutants Option:");
   menu_Printf(menu, "@p[6,5]CH   :");
   menu_Printf(menu, "@p[6,32]%% Volume");
   menu_Printf(menu, "@p[7,7]4");
   menu_Printf(menu, "@p[9,5]CO   :");
   menu_Printf(menu, "@p[9,32]%% Volume");
   menu_Printf(menu, "@p[10,7]2");
   menu_Printf(menu, "@p[12,5]NMOC :");
   menu_Printf(menu, "@p[12,32]ppm Volume");
   menu_Printf(menu, "@p[4,29]@f[@[15,#]]",
     &air_toxic_choice, &myselect_funcs);
   menu_Printf(menu, "@p[6,12]@f[@[10,#]]",
     &(scratch_model_parms.ch4), &susdbl_funcs);
   menu_Printf(menu, "@p[9,12]@fp[@[10,#]]",
     &(scratch_model_parms.co2), &susdbl_funcs);
   menu_Printf(menu, "@p[12,12]@f[@[10,#]]",
     &(scratch_model_parms.nmoc), &susdbl_funcs);

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, menu_color, menu_color, field_color);
   sed_SetPosition(sed, 5, 16);

   sed_Push(sed);
   sed_Repaint(sed);
   show_key_help(&key_menu, &key_sed, chemcomp_help_string);

/* ================================================================== */
/*                          The Main Control Loop                     */
/*                 The loop terminates on ESC or F10.                 */
/* ================================================================== */

   do
      {

/* ================================================================== */
/*   Accept field entry using Cscapes routine.  Since the special keys    */
/*  have been redefined for this program, if exit occurs, check the exit  */
/*  key to determine if the exit was caused by a special key.             */
/* ================================================================== */
```

```
      ret = sed_Go(sed);

      scratch_model_parms.include_air_toxics
                  = select_number(air_toxic_choice);

      if (old_include_at != scratch_model_parms.include_air_toxics)
         {
         warn_about_nmoc_default();
         old_include_at = scratch_model_parms.include_air_toxics;
         }

      if (ret)
         {
         if (verify_chem_comp(scratch_model_parms) == FN10)
            {
            scratch_model_parms.co2 = 100.0 - scratch_model_parms.ch4;

            sed_RepaintFields(sed);
            }
         else
            sed_GotoField(sed, 1);
         }


      switch (ret)
         {
/* ================================================================== */
/*  F1 - Help request.                                                */
/*                                                                    */
/*  Call the centralized help function (my facility - not Cscape's).  */
/* ================================================================== */
         case FN1:
            compile_help(EDIT_CHEM_COMP);
            sed_SetBaton(sed, -1);
            break;

/* ================================================================== */
/*  F2 - Reset values to their default settings.                      */
/* ================================================================== */
         case FN2:
            scratch_model_parms.include_air_toxics
                     = select_number(air_toxic_choice);
            reset_defaults(&scratch_model_parms, sed_GetFieldNo(sed));

            sed_RepaintFields(sed);
            break;
         }

/* ================================================================== */
/*  If they are trying to accept entry with an F10, verify the data first */
/*  before allowing them to leave.  If a value is bad, redefine ret to    */
/*  another value before return.  If all values are OK, return an F10 and */
/*  let the user exit.                                                */
/* ================================================================== */

      if (ret == FN10) ret = verify_chem_comp(scratch_model_parms);
      }
   while ((ret) && (ret != FN10));

/* ================================================================== */
/*                          End of Main Loop                          */
/* ================================================================== */

/* ================================================================== */
/*  Get their choice for inclusion of air toxics and write it into the    */
/*  scratch area.                                                     */
```



```
/* ================================================================== */

   scratch_model_parms.include_air_toxics
                  = select_number(air_toxic_choice);

/* ================================================================== */
/*  If the accept key was pressed, the user accepted the data and it was  */
/*  verified.  Copy the data into the original structure.             */
/* ================================================================== */
   if (ret == FN10)
      copy_model_parms(model_info, &scratch_model_parms);

/* ================================================================== */
/*                  Done now.  Destroy the menu and exit              */
/* ================================================================== */
   destroy_key_help(key_menu, key_sed);
   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);

   susdbl_funcs.fkey = old_fkey;

   return(ret);
}


/* ================================================================== */
/*      This function copies of the model parm structures to another  */
/* ================================================================== */

int copy_model_parms(dest, source)
struct model_parms *dest, *source;
{
   dest->l0 = source->l0;
   dest->k = source->k;
   dest->nmoc = source->nmoc;
   dest->ch4 = source->ch4;
   dest->co2 = source->co2;
   dest->include_air_toxics = source->include_air_toxics;
   dest->air_toxics = source->air_toxics;
   dest->number_air_toxics = source->number_air_toxics;
}


/* ================================================================== */
/*  This function resets the requested value to its default state.  Most  */
/*  the value passed in is a Cscape field number which identifies the     */
/*  parameter to be changed.                                          */
/* ================================================================== */

int reset_defaults(data, number)
struct model_parms *data;
int number;
{
   switch (number)
      {
      case 3:
         if (config.default_mode == AP_MODE)
            {
            if (data->include_air_toxics == CODISPOSAL)
               data->nmoc = DEFAULT_AP42_NMOC_CODIS;
            else data->nmoc = DEFAULT_AP42_NMOC_NOCO;
            }
         else data->nmoc = DEFAULT_REGULATORY_NMOC;
```

```
         break;

      case 1:
         data->ch4 = DEFAULT_CH4;
         break;

      case 2:
         data->co2 = DEFAULT_CO2;
         break;
      }
}

/* ========================================================================== */
/*  This function checks the data for inconsistencies.  If the value is      */
/*  incorrect, a box informs the user of the error and this function          */
/*  returns a value other than F10, the accepting code.                       */
/* ========================================================================== */

int verify_chem_comp(data)
struct model_parms data;
{
/* ========================================================================== */
/*  Do the sums of the percentages of methane and carbon dioxide yield a      */
    result greater than one?                                                  */
/* ========================================================================== */

   if (data.ch4 > 100.0)
      {
      verify_error(7);
      return(1);
      }

/* ========================================================================== */
/*         Is the methane percentage equal to (or less than) zero?           */
/*  Special Note:  this model assumes the landfill produces methane and      */
/*  cannot be used if it does not.                                            */
/* ========================================================================== */
   if (data.ch4 <= 0.0)
      {
      verify_error(9);
      return(1);
      }

   if (data.nmoc > 1E6)
      {
      verify_error(13);
      return 1;
      }

/* ========================================================================== */
/*                 Data checks!  Return the accept code.                      */
/* ========================================================================== */
   return(FN10);
}
```

```
/* ========================================================================== */
/*                                  CLOSURE.C                                 */
/* ========================================================================== */
/*  This file contains the code for displaying and altering the landfill      */
/*  closure date determined from the period data.                             */
/* ========================================================================== */
/*                             Created under Contract                         */
/*                                       to                                   */
/*                               EPA/AEERL/GECD/EMB                           */
/*                                                                            */
/*                                       by                                   */
/*                                                                            */
/*                               Radian Corporation                           */
/*                            Research Triangle Park, NC                      */
/* ========================================================================== */
/* ========================================================================== */
/*                             Revision History                               */
/* ========================================================================== */
/*    07/03/90   rlb       Initial Implementation                             */
/* ========================================================================== */

#include <stdio.h>
#include <cscape.h>
#include <scancode.h>
#include "landfill.h"
#include "landuser.h"
#include "defaults.h"
#include "helpasgn.h"
#include "colors.h"

int display_closure_year(data)
struct landfill_parms *data;
{
menu_type   menu, key_menu;
sed_type    sed, key_sed;
int         ret;
double scratch_accept_rate;
int scratch_year_closed;

/* ========================================================================== */
/*  Make a copy of the landfill closure parameters so we can change them      */
/*  without any problems.                                                     */
/* ========================================================================== */

   scratch_accept_rate = data->last_rate;
   scratch_year_closed = data->year_closed;

/* ========================================================================== */
/*                          Define the Cscape Menu                            */
/* ========================================================================== */

   menu = menu_Open();

   menu_Printf(menu, "@p[0,0]@c[%c]+@[44,-]+", menu_color);
   menu_Printf(menu, "@p[1,0]¦");
   menu_Printf(menu, "@p[1,45]¦");
   menu_Printf(menu, "@p[2,0]¦");
   menu_Printf(menu, "@p[2,9]For the period data entered,");
   menu_Printf(menu, "@p[2,45]¦");
   menu_Printf(menu, "@p[3,0]¦ the expected year of closure is determined ¦");
   menu_Printf(menu, "@p[4,0]¦");
   menu_Printf(menu, "@p[4,20]to be");
   menu_Printf(menu, "@p[4,45]¦");
   menu_Printf(menu, "@p[5,0]¦");
   menu_Printf(menu, "@p[5,45]¦");
   menu_Printf(menu, "@p[6,0]¦");
```



```
   menu_Printf(menu, "@p[6,45]¦");
   menu_Printf(menu, "@p[7,0]¦");
   menu_Printf(menu, "@p[7,45]¦");
   menu_Printf(menu, "@p[8,0]¦    assuming a refuse acceptance rate of    ¦");
   menu_Printf(menu, "@p[9,0]¦");
   menu_Printf(menu, "@p[9,45]¦");
   menu_Printf(menu, "@p[10,0]¦");
   menu_Printf(menu, "@p[10,29]Mg/year");
   menu_Printf(menu, "@p[10,45]¦");
   menu_Printf(menu, "@p[11,0]¦");
   menu_Printf(menu, "@p[11,45]¦");
   menu_Printf(menu, "@p[12,0]¦     from current year to closure year.     ¦");
   menu_Printf(menu, "@p[13,0]¦");
   menu_Printf(menu, "@p[13,45]¦");
   menu_Printf(menu, "@p[14,0]+@{44,-}+");
   menu_Printf(menu, "@p[6,20]@f[####]",
     &scratch_year_closed, &susint_funcs);
   menu_Printf(menu, "@p[10,13]@fp[@[14,#]]",
     &scratch_accept_rate, &susdbl_funcs);

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, menu_color, menu_color, field_color);
   sed_SetPosition(sed, 5, 17);

   sed_Push(sed);

   sed_Repaint(sed);

/* ========================================================================== */
/*  If the closure year has already been encountered, the screen is info      */
/*  only and cannot be changed so it doesn't need help.                       */
/* ========================================================================== */

   if (data->refuse_in_place[data->number_of_periods - 1] < data->capacity)
      show_key_help(&key_menu, &key_sed, generic_help_string);
   else
      show_key_help(&key_menu, &key_sed, continue_help_string);

/* ========================================================================== */
/*  If the closure date has yet to be encountered, allow the user to edit     */
/*  the computer generated value.  First, the closure date was already        */
/*  determined from periods.c.  Recalculate the last rate using the           */
/*  closure date.  This has the affect of making the last rate an exact       */
/*  average rate to reach closure.  Then, activate the field entry loop.      */
/* ========================================================================== */

   if (data->refuse_in_place[data->number_of_periods - 1] < data->capacity)
      {
      calculate_acceptance_rate(data, scratch_year_closed,
            &scratch_accept_rate);

      sed_RepaintFields(sed);

      do
         {
         ret = sed_Go(sed);

         switch (ret)
            {
/* ========================================================================== */
/*  F1 - Help requested                                                       */
/*                                                                            */
/*  Call the centralized help facility.                                       */
/* ========================================================================== */
```

```
            case FN1:
               compile_help(DISPLAY_CLOSURE_YEAR);
               sed_SetBaton(sed, -1);
               break;

/* ========================================================================== */
/*  F2 - Reset the closure year to its default value                          */
/* ========================================================================== */
            case FN2:
               scratch_year_closed = data->year_closed;
               break;
            }

/* ========================================================================== */
/*  If ret is non-zero, check to make sure the closure date listed is a       */
/*  valid entry.  If so, recalculate the acceptance rate assuming the new     */
/*  closure date and redisplay.                                               */
/* ========================================================================== */

         if (ret)
            {

#ifdef DEBUG
            printf("Validating closure data\n");
            getchar();
#endif

            if (! validate_closure_data(data, &scratch_year_closed))
               {
               calculate_acceptance_rate(data, scratch_year_closed,
                                         &scratch_accept_rate);

               sed_RepaintFields(sed);
               }
            else
               ret = -1;
            }
         }
      while ((ret) && (ret != FN10));

/* ========================================================================== */
/*  Since the data was verified on the fly and F10 was pressed, the user      */
/*  has accepted the closure date displayed.  Update the original struct.     */
/* ========================================================================== */

      if (ret == FN10)
         {
         data->year_closed = scratch_year_closed;
         data->last_rate = scratch_accept_rate;
         }
      }

/* ========================================================================== */
/*                      End editable closure year case                        */
/* ========================================================================== */

   else

/* ========================================================================== */
/*  The closure date has already been encountered so the user shouldn't       */
/*  change the closure date.  Just watch for a key and terminate.             */
/* ========================================================================== */

      kb_Read();

   destroy_key_help(key_menu, key_sed);
```



```
   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);
   return(ret);
}


/* ========================================================================== */
/*  This function checks to see if the date entered is a legitimate value    */
/* ========================================================================== */

int validate_closure_data(data, year)
struct landfill_parms *data;
int *year;
{
/* ========================================================================== */
/*         Is the closure year greater than the current year?                 */
/*  Since the calling function goes another path if the closure year has      */
/*  been encountered, this is not a legitimate entry.                         */
/* ========================================================================== */

   if (data->current_year >= *year)
      {
      verify_error(5);
      return(1);
      }

/* ========================================================================== */
/*           Does the year exceed the program design limits?                  */
/* ========================================================================== */

   if (*year - data->year_opened > MAX_DURATION)
      {
      verify_error(6);
      return(1);
      }

/* ========================================================================== */
/*                            Year OK - return zero                           */
/* ========================================================================== */

   return(0);
}


/* ========================================================================== */
/*  This function calculates the acceptance rate given the closure year.      */
/*  This is a two step process originating with periods.c.  First, the        */
/*  closure year is calculated based on the last observed (or logically       */
/*  interpolated) acceptance rate.  The capacity is reduced by whole          */
/*  numbers of last rates until the remainder is less than the last rate.     */
/*  The closure year is therefore the whole number of last rates required     */
/*  with one more added if the remainder is non-zero.  Consequently, the      */
/*  last rate is not a uniform value over the years and makes computing       */
/*  the last rate displayed in the closure screen inexact.  This function     */
/*  is a logical companion to the one in periods and it rationalizes the      */
/*  two rates.                                                                */
/* ========================================================================== */

int calculate_acceptance_rate(data, close, rate)
struct landfill_parms *data;
int close;
double *rate;
{
int remaining, i;
```

```
/* ========================================================================== */
/*  First, determining the number of remaining years and the remaining       */
/*  refuse until capacity is reached.  Calculate the acceptance rate and     */
/*  return it.                                                               */
/* ========================================================================== */

   remaining = close - data->current_year;
   i = data->number_of_periods - 1;

   if (remaining <= 0)
      {
      *rate = 0.0;
      return(0);
      }

   *rate = (data->capacity - data->refuse_in_place[i]) / remaining;
}
```

```
/* ========================================================================== */
/*                                  CONFIG.C                                  */
/* ========================================================================== */
/*  This file contains the code for displaying and altering the landfill      */
/*  program configuration settings.                                           */
/* ========================================================================== */
/*                            Created under Contract                          */
/*                                      to                                    */
/*                              EPA/AEERL/GECD/EMB                            */
/*                                                                            */
/*                                      by                                    */
/*                                                                            */
/*                              Radian Corporation                            */
/*                           Research Triangle Park, NC                       */
/* ========================================================================== */
/* ========================================================================== */
/*                             Revision History                               */
/* ========================================================================== */
/*    07/03/90   rlb       Initial Implementation                             */
/* ========================================================================== */

#include <stdio.h>
#include <cscape.h>
#include <scancode.h>
#include <string.h>
#include "landfill.h"
#include "landuser.h"
#include "defaults.h"
#include "helpasgn.h"
#include "colors.h"

/* ========================================================================== */
/*  This function handles provides the menu for changing and writing the      */
/*  landfill program configuration settings to disk.                          */
/* ========================================================================== */

/* ========================================================================== */
/*  First, define the device settings for the toggle selection function.      */
/* ========================================================================== */

static char *devices[] = {"LPT1", "LPT2", "COM1", "COM2", NULL};
static char *defaults[] = {"REGULATORY", "AP-42", NULL};

int configure_system()
{
FILE *config_file;
menu_type   menu, key_menu;
sed_type    sed, key_sed;
struct select_struct device_choice;
struct select_struct defaults_choice;
int ret, len, i;
int scratch_limit;
char scratch_dir[265];
int verify_ypc(int years);

/* ========================================================================== */
/* Set the initial value of the scratch years passed closure.                 */
/* ========================================================================== */

   scratch_limit = config.years_passed_closure;


/* ========================================================================== */
/*                          Build the device choices.                         */
/* ========================================================================== */
```



```
   device_choice.number = config.print_device;
   device_choice.list = devices;

/* ========================================================================== */
/*                          Build the defaults choices.                       */
/* ========================================================================== */


   defaults_choice.number = config.default_mode;
   defaults_choice.list = defaults;


/* ========================================================================== */
/*                 Build the working directory ticker string.                 */
/* ========================================================================== */

   tick_SetLen(scratch_dir, 254);
   tick_SetOffset(scratch_dir, 0);

   len = strlen(config.working_dir);
   if (*(config.working_dir + len - 1) != '\\')
      {
      *(config.working_dir + len) = '\\';
      *(config.working_dir + len + 1) = '\0';
      }

   strcpy(tick_GetString(scratch_dir), config.working_dir);

/* ========================================================================== */
/*                               Define the Cscape Menu                       */
/* ========================================================================== */

   reset_insert_status();

   menu = menu_Open();

   menu_Printf(menu, "@p[0,0]@c[%c]+@[48,-]+", menu_color);
   menu_Printf(menu, "@p[1,0]¦");
   menu_Printf(menu, "@p[1,13]Edit System Configuration");
   menu_Printf(menu, "@p[1,49]¦");
   menu_Printf(menu, "@p[2,0]¦@[48,-]¦");
   menu_Printf(menu, "@p[3,0]¦");
   menu_Printf(menu, "@p[3,49]¦");
   menu_Printf(menu, "@p[4,0]¦ Working Directory :");
   menu_Printf(menu, "@p[4,49]¦");
   menu_Printf(menu, "@p[5,0]¦");
   menu_Printf(menu, "@p[5,49]¦");
   menu_Printf(menu, "@p[6,0]¦ Default Print Device :");
   menu_Printf(menu, "@p[6,49]¦");
   menu_Printf(menu, "@p[7,0]¦");
   menu_Printf(menu, "@p[7,49]¦");
   menu_Printf(menu, "@p[8,0]¦ Defaults Mode :");
   menu_Printf(menu, "@p[8,49]¦");
   menu_Printf(menu, "@p[9,0]¦");
   menu_Printf(menu, "@p[9,49]¦");
   menu_Printf(menu, "@p[10,0]¦ Calculation Years Passed Closure: ");
   menu_Printf(menu, "@p[10,49]¦");
   menu_Printf(menu, "@p[11,0]¦");
   menu_Printf(menu, "@p[11,49]¦");
   menu_Printf(menu, "@p[12,0]+@[48,-]+");
   menu_Printf(menu, "@p[4,22]@f[@[26,#]]", scratch_dir, &ticker_funcs);
   menu_Printf(menu, "@p[6,25]@f[######]", &device_choice, &select_funcs);
   menu_Printf(menu, "@p[8,18]@f[@[11,#]]", &defaults_choice, &select_funcs);
   menu_Printf(menu, "@p[10,37]@f[###]", &scratch_limit, &susint_funcs);

   menu_Flush(menu);
```

```
   sed = sed_Open(menu);
   sed_SetColors(sed, menu_color, menu_color, field_color);
   sed_SetPosition(sed, 5, 15);

   sed_Push(sed);
   sed_Repaint(sed);
   show_key_help(&key_menu, &key_sed, config_help_string);

/* ========================================================================== */
/*                      This is the Main Control Loop                         */
/*                    This loop terminates on ESC or F10.                     */
/* ========================================================================== */

   do
      {
      do
         {
/* ========================================================================== */
/*                          Handle the field entry.                           */
/* ========================================================================== */

         ret = sed_Go(sed);

/* ========================================================================== */
/*  Help Requested - Call centralized help function                           */
/* ========================================================================== */

         if (ret == FN1)
            {
            compile_help(CONFIGURE_SYSTEM);
            sed_SetBaton(sed, -1);
            }
         }
      while ((ret) && (ret != FN10));
      }
   while (verify_ypc(scratch_limit));


/* ========================================================================== */
/*                              End Main Loop                                 */
/* ========================================================================== */

/* ========================================================================== */
/*  If the FN10 key was pressed, the user accepted the current status of      */
/*  the configuration variables.  Copy the scratch variables back to the      */
/*  original ones and write them out to the configuration file.               */
/* ========================================================================== */

   if (ret == FN10)
      {

/* ========================================================================== */
/*  Cscape provides a macro interface to their ticker string that returns    */
/*  the pointer to the string part of the ticker string.  This is padded      */
/*  with spaces and must be truncated.                                        */
/* ========================================================================== */
      strcpy(config.working_dir, tick_GetString(scratch_dir));

      len = strlen(config.working_dir);

      for (i = 0 ; i < len ; i++)
         if (config.working_dir[i] == ' ')
            {
            config.working_dir[i] = '\0';
            break;
```



```
            }

      config.print_device = device_choice.number;
      config.default_mode = defaults_choice.number;

      if (! verify_ypc(scratch_limit))
         config.years_passed_closure = scratch_limit;

      len = strlen(config.working_dir);

/* ========================================================================== */
/*  The convention chosen is to make sure all directory paths do not end      */
/*  in a backslash.  In this function, the backslash is put on when           */
/*  first entering and taken off here.  (Also, this code is repeated          */
/*  this if is finish but there is no backslash for it to find so this        */
/*  is OK.  I need it formatted correctly when I write it to disk.            */
/* ========================================================================== */

      if (*(config.working_dir + len - 1) == '\\')
         *(config.working_dir + len - 1) = '\0';

/* ========================================================================== */
/*                        One of the misc_utl functions.                      */
/* ========================================================================== */
      capital_string(config.working_dir);

/* ========================================================================== */
/*  Piece the filename together and write the configuration information       */
/*  to LANDFILL.CFG.                                                          */
/* ========================================================================== */

      strcpy(scratch_dir, config.landfill_dir);
      strcat(scratch_dir, "LANDFILL.CFG");

      config_file = fopen(scratch_dir, "w");

      if (config_file != NULL)
         {
         fprintf(config_file, "%s\n", config.working_dir);
         fprintf(config_file, "%d\n", config.print_device);
         fprintf(config_file, "%d\n", config.default_mode);
         fprintf(config_file, "%d\n", config.years_passed_closure);

         fclose(config_file);
         }
      }

   len = strlen(config.working_dir);

   if (*(config.working_dir + len - 1) == '\\')
      *(config.working_dir + len - 1) = '\0';

/* ========================================================================== */
/*                    All done!  Delete the menu and return.                  */
/* ========================================================================== */

   destroy_key_help(key_menu, key_sed);
   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);
   return(ret);
}
```

```
int verify_ypc(int years)
{
   if (years > MAX_YEARS_PAST_CLOSURE)
      {
      verify_error(12);
      return(1);
      }

   return(0);
}
```

```
/* ========================================================================== */
/*                                  CREDITS.C                                 */
/* ========================================================================== */
/*  This file contains the code for displaying the title screen for the       */
/*  landfill program.                                                         */
/* ========================================================================== */
/*                              Created under Contract                        */
/*                                      to                                    */
/*                                EPA/AEERL/GECD/EMB                          */
/*                                                                            */
/*                                      by                                    */
/*                                                                            */
/*                                Radian Corporation                          */
/*                             Research Triangle Park, NC                     */
/* ========================================================================== */
/* ========================================================================== */
/*                                Revision History                            */
/* ========================================================================== */
/*    07/03/90   rlb       Initial Implementation                             */
/* ========================================================================== */

#include <stdio.h>
#include <cscape.h>
#include <string.h>
#include "landfill.h"
#include "helpsgn.h"
#include "colors.h"

/* ========================================================================== */
/*         This function defines and displays the title screen.               */
/* ========================================================================== */

void credits()
{
menu_type   menu, key_menu;
sed_type    sed, key_sed;
int old_cursor;
char version_string[35];

   old_cursor = vid_GetCursorType();
   vid_SetCursorType(CURSOR_NONE);

   strcenter(strcpy(version_string, TITLE_STRING), 33);

   menu = menu_Open();

   menu_Printf(menu, "@p[0,0]@c[%c]+@[33,-]+", menu_color);
   menu_Printf(menu, "@p[1,0]¦");
   menu_Printf(menu, "@p[1,34]¦");
   menu_Printf(menu, "@p[2,0]¦");
   menu_Printf(menu, "@p[2,13]Landfills");
   menu_Printf(menu, "@p[2,34]¦");
   menu_Printf(menu, "@p[3,0]¦");
   menu_Printf(menu, "@p[3,11]Air Emissions");
   menu_Printf(menu, "@p[3,34]¦");
   menu_Printf(menu, "@p[4,0]¦");
   menu_Printf(menu, "@p[4,10]Estimation Model");
   menu_Printf(menu, "@p[4,34]¦");
   menu_Printf(menu, "@p[5,0]¦");
   menu_Printf(menu, "@p[5,34]¦");
   menu_Printf(menu, "@p[6,0]¦");
   menu_Printf(menu, "@p[6,34]¦");
   menu_Printf(menu, "@p[7,0]¦@fp[@[33,#]]¦", version_string, &string_funcs);
   menu_Printf(menu, "@p[8,0]¦");
   menu_Printf(menu, "@p[8,34]¦");
   menu_Printf(menu, "@p[9,0]¦");
```



```
   menu_Printf(menu, "@p[9,34]¦");
   menu_Printf(menu, "@p[10,0]+@[33,-]+");

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, menu_color, menu_color, menu_color);
   sed_SetPosition(sed, 7, 22);

   sed_Push(sed);
   sed_Repaint(sed);
   show_key_help(&key_menu, &key_sed, any_key_string);

   kb_Read();

   destroy_key_help(key_menu, key_sed);
   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);

   vid_SetCursorType(old_cursor);

   return;
}
```

```
/* ========================================================================== */
/*                                 DISPRSLT.C                                 */
/* ========================================================================== */
/*  This file contains the code for displaying the output method              */
/*  selection menu.                                                           */
/* ========================================================================== */
/*                            Created under Contract                          */
/*                                      to                                    */
/*                              EPA/AEERL/GECD/EMB                            */
/*                                                                            */
/*                                      by                                    */
/*                                                                            */
/*                              Radian Corporation                            */
/*                           Research Triangle Park, NC                       */
/* ========================================================================== */
/* ========================================================================== */
/*                              Revision History                              */
/* ========================================================================== */
/*    07/03/90   rlb        Initial Implementation                            */
/* ========================================================================== */

/* ========================================================================== */
/*  This function defines and presents the menu to allow the user to          */
/*  select the report method : tabular or on-screen graphic.                  */
/* ========================================================================== */

#include <stdio.h>
#include <cscape.h>
#include <scancode.h>
#include "helpasgn.h"
#include "colors.h"

int display_results_menu(choice)
int choice;
{
menu_type   menu, key_menu;
sed_type    sed, key_sed;
int         ret;

   menu = menu_Open();

   buildbox(menu, menu_color, 10, 29);
   menu_Printf(menu, "@p[1,4]Display Results Menu");
   menu_Printf(menu, "@p[4,1]@f[      Tabular Report       ]",
     NULL, &menu_funcs);
   menu_Printf(menu, "@p[5,1]@f[    On-screen Graphic      ]",
     NULL, &menu_funcs);
   menu_Printf(menu, "@p[7,1]@f[       Previous Menu       ]",
     NULL, &menu_funcs);

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, menu_color, menu_color, field_color);
   sed_SetPosition(sed, 7, 25);

   sed_Push(sed);

   sed_GotoField(sed, choice);

   sed_Repaint(sed);
   show_key_help(&key_menu, &key_sed, generic_menu_help_string);

/* ========================================================================== */
/*                                 Main Loop                                  */
/*  This loop continues until the ret code from sed_Go is less than the       */
```



```
/*  field count.  sed_Go returns the menu field number of the item that       */
/*  generated the exit, the value of a special exit key, or 0 if ESC was      */
/*  pressed.                                                                  */
/* ========================================================================== */

   do
      {
/* ========================================================================== */
/*                              Get the field input                           */
/* ========================================================================== */
      ret = sed_Go(sed);

/* ========================================================================== */
/*        Call the centralize help facility if the FN1 was pressed.           */
/* ========================================================================== */

      if (ret == FN1)
         {
         compile_help(DISPLAY_RESULTS_MENU);
         sed_SetBaton(sed, -1);
         }
      }
   while (ret > sed_GetFieldCount(sed));

   destroy_key_help(key_menu, key_sed);
   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);
   return(ret);
}
```

```
/* ========================================================================== */
/*                                  DO_GRAPH.C                                */
/* ========================================================================== */
/*  This file contains the code for displaying the results graphically        */
/* ========================================================================== */
/*                             Created under Contract                         */
/*                                      to                                    */
/*                               EPA/AEERL/GECD/EMB                           */
/*                                                                            */
/*                                      by                                    */
/*                                                                            */
/*                                Radian Corporation                          */
/*                             Research Triangle Park, NC                     */
/* ========================================================================== */
/* ========================================================================== */
/*                                Revision History                            */
/* ========================================================================== */
/*    07/03/90   rlb        Initial Implementation                            */
/* ========================================================================== */

#include <stdio.h>
#include <cscape.h>
#include <malloc.h>
#include <math.h>
#include <ctype.h>
#include <graph.h>
#include <pgchart.h>
#include <graph.h>
#include "landfill.h"
#include "helpasgn.h"

#define WORK_SPACE 15

char *legend1 = "Estimated Emissions";
char *legend2 = "Projected Emissions";

/* ========================================================================== */
/*              This function actually displays the graphic                   */
/* ========================================================================== */

int display_onscreen_graph(x, y, current, count, title, units)
int *x, count, current;
double *y;
char *title, *units;
{
double max_y, factor, map_x, map_y;
struct videoconfig screen;
chartenv env, titles;
int i, j = 0, power, code, current_index;
int lx1, lx2, ly1, ly2;

float *y_copy, *x_copy, range, base;

/* ========================================================================== */
/*     Create local copies of the data compatible with the presentation       */
/*                          graphics functions                                */
/* ========================================================================== */

   x_copy = (float *) calloc((size_t) count, sizeof(float));
   y_copy = (float *) calloc((size_t) count, sizeof(float));

   if ((x_copy == NULL) || (y_copy == NULL))
      show_mem_error("display_onscreen_graph", "xy_data");

/* ========================================================================== */
/*  Determine the maximum y value.  This will be used to autoscale the y      */
```



```
/*  axis.                                                                     */
/* ========================================================================== */

   for (max_y = y[0], i = 1 ; i < count ; i++)
      if (max_y < y[i]) max_y = y[i];

   if (max_y > 0.0000)
      {
      power = floor(log10(max_y)) - 1;
      factor = pow(10.0, (double) power);
      }
   else
      {
      factor = 1;
      power = 0;
      }

   current_index = current - x[0];

   for (i = 0 ; i < count ; i++)
      {
      x_copy[i] = (float) x[i];
      y_copy[i] = (float) y[i] / factor;
      }

/* ========================================================================== */
/*  Set the video mode to graphics.  If the video card does not allow         */
/*  an acceptable mode, return to the calling function.                       */
/* ========================================================================== */

   if (! set_video_mode(&screen)) return(1);

/* ========================================================================== */
/*                        Set up presentation graphics                        */
/* ========================================================================== */

   _pg_initchart();
   _pg_defaultchart(&env, _PG_SCATTERCHART, _PG_POINTANDLINE);

/* ========================================================================== */
/*   Turn off autoscale after analyzing so I can take care of some of         */
/*   the fundamental things myself.                                           */
/* ========================================================================== */

   env.yaxis.autoscale = env.xaxis.autoscale = FALSE;

/* ========================================================================== */
/*                Set up some of the basic scale definitions                  */
/* ========================================================================== */

   env.xaxis.grid = env.yaxis.grid = FALSE;
   env.xaxis.labeled = env.yaxis.labeled = TRUE;
   env.xaxis.rangetype = env.yaxis.rangetype = _PG_LINEARAXIS;
   env.xaxis.ticformat = env.yaxis.ticformat = _PG_DECFORMAT;

/* ========================================================================== */
/*   Determine the x scale extent, set the tick marks, and set the axis       */
/*   scale factor.                                                            */
/* ========================================================================== */

/*   env.xaxis.scalemax = (float)((((int) x_copy[count - 1] / 10) * 10) + 10); *
/

   range = x_copy[count - 1] - x_copy[0];
   base = range / pow(10.0, floor(log10(range)));
```

```
   if (base < 1.5) base = 1.0;
   if ((base >= 1.5) && (base < 3.0)) base = 2.0;
   if (base >= 3.0) base = 5.0;

   env.xaxis.ticinterval = base * pow(10.0, floor(log10(range)) - 1);

   env.xaxis.scalemin = floor((int) x_copy[0] / base) * base;
        env.xaxis.scalemax = ceil((double) x_copy[count - 1] / base) * base;

   env.xaxis.scalefactor = 1.0;
   env.xaxis.scaletitle.title[0] = '\0';

/* ================================================================== */
/*   Determine the y scale extent, set the tick marks, and set the axis  */
/*   scale factor.                                                      */
/* ================================================================== */

   env.yaxis.scalemin = 0.0;
   env.yaxis.scalemax = (float)((int)(max_y / factor) + 1);

   range = (env.yaxis.scalemax - env.yaxis.scalemin);

   base = range / pow(10.0, floor(log10(range)));

   if (base < 1.5) base = 1.0;
   if ((base >= 1.5) && (base < 3.5)) base = 2.0;
   if (base >= 3.5) base = 5.0;

   env.yaxis.ticinterval = base;

   env.yaxis.scalemax = ceil(env.yaxis.scalemax / base) * base;

   env.yaxis.scalefactor = 1;

   strcpy(env.xaxis.axistitle.title, "Year (* = Current Year)");

   sprintf(env.yaxis.axistitle.title, "Emission Rate");
        sprintf(env.yaxis.scaletitle.title, "1E%-2d %s", power, units);

   env.xaxis.axistitle.justify = env.yaxis.axistitle.justify =
      _PG_CENTER;

   strcpy(env.maintitle.title, "Projected ");
   strcat(env.maintitle.title, title);
   for (i = strlen(env.maintitle.title) - 1 ; i >= 0 ; i--)
      if (isalnum(env.maintitle.title[i]))
         {
         env.maintitle.title[i + 1] = '\0';
         break;
         }

   env.xaxis.ticdecimals = 0;
   env.yaxis.ticdecimals = 1;

   strcat(env.maintitle.title, " Emission Rate");

/* ================================================================== */
/*  First, analyze the chart.  This allows presentation graphics to set  */
/*  the chart parameters for me.  I reduce the chart space a little to   */
/*  allow a legend later.                                              */
/* ================================================================== */

        code = _pg_analyzescatter(&env, (float far *) (x_copy + current_index),
                                  (float far *) (y_copy + current_index), 1);

        env.chartwindow.border = 0;
```



```
        env.datawindow.border = 0;

        env.chartwindow.y2 -= WORK_SPACE;

/* ================================================================== */
/*  Now, draw the chart.  The only point I want explicitly place on the  */
/*  graph is the current year (labeled with a *).  I will take care of   */
/*  plotting the rest of the data.                                     */
/* ================================================================== */

   code = _pg_chartscatter(&env, (float far *) (x_copy + current_index),
                                 (float far *) (y_copy + current_index), 1);

/* ================================================================== */
/*  Use the information returned in the chartenv structure to determine  */
/*  the scale.                                                         */
/* ================================================================== */

   map_x = (double) (env.datawindow.x2 - env.datawindow.x1)
                    / (double) (env.xaxis.scalemax - env.xaxis.scalemin);
   map_y = (double) (env.datawindow.y1 - env.datawindow.y2)
                    / (double) (env.yaxis.scalemax - env.yaxis.scalemin);

/* ================================================================== */
/*       Plot the estimated emissions line in color 1 and a solid line   */
/* ================================================================== */

   _setlinestyle(0xFFFF);

        _setcolor(1);

        _moveto((short) (env.datawindow.x1 +
                ((x_copy[0] - env.xaxis.scalemin) * map_x)),
                   (short)(env.datawindow.y2 + (map_y * y_copy[0])));

   for (i = 1 ; i <= current_index ; i++)
      _lineto((short) (env.datawindow.x1 +
                (int) (map_x * (x_copy[i] - env.xaxis.scalemin))),
                   (short) (env.datawindow.y2 + (int) (map_y * y_copy[i])));

/* ================================================================== */
/*       Plot the projected emissions line in color 2 and a dotted line   */
/* ================================================================== */

        _setcolor(2);

   _setlinestyle(0x5555);

   for (i = current_index + 1 ; i < count; i++)
      _lineto((short) (env.datawindow.x1 +
                (int) (map_x * (x_copy[i] - env.xaxis.scalemin))),
                   (short) (env.datawindow.y2 + (int) (map_y * y_copy[i])));

/* ================================================================== */
/*  Put a legend on the chart.  In order to use the graphics font used by */
/*  presentation graphics and prevent clipping, this requires creating a  */
/*  chart with presentation graphics.                                  */
/* ================================================================== */

   _pg_defaultchart(&titles, _PG_SCATTERCHART, _PG_POINTANDLINE);

        lx1 = titles.chartwindow.x1 = env.chartwindow.x1;
        lx2 = titles.chartwindow.x2 = env.chartwindow.x2;
        ly1 = titles.chartwindow.y1 = env.chartwindow.y2;
        ly2 = titles.chartwindow.y2 = env.chartwindow.y2 + WORK_SPACE;
```

```
        _pg_hlabelchart(&titles, lx2 / 2 - 8 * (2 + strlen(legend1)),
                  5, 1, legend1);

        _pg_hlabelchart(&titles, lx2 / 2 + 48, 5, 1, legend2);

   _setlinestyle(0xFFFF);
        _setcolor(1);

   _moveto((short)(lx2 / 2 - 8 * (4 + strlen(legend1))), ly1 + 9);
   _lineto((short)(lx2 / 2 - 8 * (8 + strlen(legend1))), ly1 + 9);

   _setlinestyle(0x5555);
        _setcolor(2);

   _moveto((short)(lx2 / 2 + 32), ly1 + 9);
   _lineto((short)(lx2 / 2), ly1 + 9);

/* ============================================================================ */
/*          If plotting was successful, wait for a key and clean up             */
/* ============================================================================ */

   if (code != 0)
      {
      printf("Chartscatter returned %d\n", code);
      getchar();
      }
   else
      kb_Read();

   _setvideomode(_DEFAULTMODE);

   free(x_copy);
   free(y_copy);

   return(0);
}



/* ============================================================================ */
/*  This function puts the card in graphics mode.  Supported cards are          */
/*  VGA, EGA, CGA, and Hercules (with MSHERC installed).                        */
/* ============================================================================ */

int set_video_mode(screen)
struct videoconfig *screen;
{
int ret = 0;

   _getvideoconfig(screen);

   switch (screen->adapter)
      {
                case _OCGA:
                case _OEGA:
                case _OVGA:
         ret = _setvideomode(_ORESCOLOR);
         break;

      case _CGA:
                ret = _setvideomode(_HRESBW);

         break;

      case _EGA:
              ret = _setvideomode(_ERESCOLOR);
```



```
         break;

      case _SVGA:
      case _VGA:
      case _MCGA:
         ret = _setvideomode(_VRES16COLOR);
         break;

      case _HGC:
         ret = _setvideomode(_HERCMONO);
         break;

      default:
         _setvideomode(_DEFAULTMODE);
      }

   if (! ret) no_card_support();
   return(ret);
}


/* ============================================================================ */
/*       This function pops up an error if the card is not supported.          */
/* ============================================================================ */

int no_card_support()
{
menu_type   menu, key_menu;
sed_type    sed, key_sed;
int         ret;

   menu = menu_Open();

   menu_Printf(menu, "@p[0,0]@c[\x07]+@[31,-]+");
   menu_Printf(menu, "@p[1,0]|     Monitor Not Supported     |");
   menu_Printf(menu, "@p[2,0]|@[31,-]|");
   menu_Printf(menu, "@p[3,0]|");
   menu_Printf(menu, "@p[3,32]|");
   menu_Printf(menu, "@p[4,0]|   Your monitor card is not    |");
   menu_Printf(menu, "@p[5,0]|   supported by this system.   |");
   menu_Printf(menu, "@p[6,0]|");
   menu_Printf(menu, "@p[6,32]|");
   menu_Printf(menu, "@p[7,0]+@[31,-]+");

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, '\x07', '\x07', '\x70');
   sed_SetPosition(sed, 9, 23);

   sed_Push(sed);
   sed_Repaint(sed);
   show_key_help(&key_menu, &key_sed, any_key_string);

   kb_Read();

   destroy_key_help(key_menu, key_sed);
   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);
   return(ret);
}
```

```
/* ========================================================================== */
/*                                  EDITMAIN.C                                */
/* ========================================================================== */
/*  This file the major control structures for the edit portion of the        */
/*  Landfill Air Emmisions model.                                             */

/* ========================================================================== */
/*                            Created under Contract                          */
/*                                    to                                      */
/*                              EPA/AEERL/GECD/EMB                            */
/*                                                                            */
/*                                    by                                      */
/*                                                                            */
/*                              Radian Corporation                            */
/*                           Research Triangle Park, NC                       */
/* ========================================================================== */
/* ========================================================================== */
/*                              Revision History                              */
/* ========================================================================== */
/*    07/03/90   rlb       Initial Implementation                             */
/*    01/21/94   rlb       Changed include_air_toxics flag comparison to      */
/*                            NO_AT to support landfill disposal type.        */
/* ========================================================================== */

#include <stdio.h>
#include <cscape.h>
#include <scancode.h>
#include "landfill.h"
#include "defaults.h"

boolean edit_main(landfill_info, model_info, k_calc_info, calc_possible)
struct landfill_parms *landfill_info;
struct model_parms *model_info;
struct k_calc_parms *k_calc_info;
boolean *calc_possible;
{
int choice = 1, k_choice = 1, ret;
boolean changed = FALSE;

extern int edit_menu(int);
extern int air_toxics_menu(struct air_toxic_structure **, int *);
extern int edit_chem_comp(struct model_parms *);
extern int edit_periods(struct landfill_parms *);
extern int l0_selection_menu(double *);
extern int k_selection_menu(int);
extern int edit_k_value(double *);
extern int warn_about_2e();
extern int calculate_k_menu(struct k_calc_parms *, struct model_parms *);

/* ========================================================================== */
/* Essentially, this function is one big loop which activates all of the     */
/* menus coordinates the interrelations of the menus.                         */
/* ========================================================================== */

   while (choice = edit_menu(choice - 1))
      {
      switch (choice)
         {
         case 0:
            choice = 1;
            break;

         case 1:
            if (ret = edit_chem_comp(model_info))
               {
               if (model_info->include_air_toxics != NO_AT)
```



```
                  air_toxics_menu(&(model_info->air_toxics),
                                  &(model_info->number_air_toxics));
               else
                  {
                  if (model_info->air_toxics != NULL)
                     free(model_info->air_toxics);

                  model_info->air_toxics = NULL;

                  model_info->number_air_toxics = 0;
                  }
               }

            break;

         case 2:
            switch (ret = k_choice = k_selection_menu(k_choice - 1))
               {
               case 1:
                  if (config.default_mode == AP_MODE)
                     model_info->k = DEFAULT_AP42_K;
                  else model_info->k = DEFAULT_REGULATORY_K;
                  break;

               case 2:
                  ret = edit_k_value(&(model_info->k));

                  break;

               case 3:
                  if (warn_about_2e() != ESC)
                     ret = calculate_k_menu(k_calc_info, model_info);

                  break;

               default:
                  k_choice = 1;
               }
            break;

         case 3:
            ret = l0_selection_menu(&(model_info->l0));

            break;

         case 4:
            ret = edit_periods(landfill_info);
            if (ret) *calc_possible = TRUE;
            break;

         case 5:
            return(changed);
         }

      if (ret) changed = TRUE;
      }

   return(changed);
}
```

```
/* ========================================================================== */
/*                                 EDITMENU.C                                 */
/* ========================================================================== */
/*      This file contains the code for displaying edit menu                  */
/* ========================================================================== */
/*                            Created under Contract                          */
/*                                      to                                    */
/*                              EPA/AEERL/GECD/EMB                            */
/*                                                                            */
/*                                      by                                    */
/*                                                                            */
/*                              Radian Corporation                            */
/*                           Research Triangle Park, NC                       */
/* ========================================================================== */
/* ========================================================================== */
/*                             Revision History                               */
/* ========================================================================== */
/*    07/03/90   rlb       Initial Implementation                             */
/* ========================================================================== */

#include <stdio.h>
#include <cscape.h>
#include <scancode.h>
#include "helpasgn.h"
#include "colors.h"

int edit_menu(choice)
int choice;
{
menu_type   menu, key_menu;
sed_type    sed, key_sed;
int         ret;

void show_edit_help();

   menu = menu_Open();

   buildbox(menu, menu_color, 12, 36);
   menu_Printf(menu, "@p[1,10]Edit Study Data");

   menu_Printf(menu, "@p[4,1]@f[@[7, ]Chemical Composition@[7, ]]",
     NULL, &menu_funcs);
   menu_Printf(menu, "@p[5,1]@f[    Methane Rate Constant  (k)    ]",
     NULL, &menu_funcs);
   menu_Printf(menu, "@p[6,1]@f[      Methane Potential (Lo)      ]",
     NULL, &menu_funcs);
   menu_Printf(menu, "@p[7,1]@f[@[9, ]Operational Data@[9, ]]",
     NULL, &menu_funcs);
   menu_Printf(menu, "@p[9,1]@f[@[10, ]Previous Menu@[11, ]]",
     NULL, &menu_funcs);

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, menu_color, menu_color, field_color);
   sed_SetPosition(sed, 6, 22);

   sed_Push(sed);

   sed_GotoField(sed, choice);

   sed_Repaint(sed);
   show_key_help(&key_menu, &key_sed, generic_menu_help_string);

   do
      {
```



```
      ret = sed_Go(sed);

      if (ret == FN1)
         {
         compile_help(EDIT_MENU);
         sed_SetBaton(sed, -1);
         }
      }
   while (ret > sed_GetFieldCount(sed));

   destroy_key_help(key_menu, key_sed);
   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);
   return(ret);
}
```

```
/* ========================================================================= */
/*                                 EXIT.C                                    */
/* ========================================================================= */
/*  This file contains the code for displaying the exit selection menu       */
/* ========================================================================= */
/*                           Created under Contract                          */
/*                                   to                                      */
/*                             EPA/AEERL/GECD/EMB                            */
/*                                                                           */
/*                                   by                                      */
/*                                                                           */
/*                             Radian Corporation                            */
/*                          Research Triangle Park, NC                       */
/* ========================================================================= */
/* ========================================================================= */
/*                           Revision History                                */
/* ========================================================================= */
/*    07/03/90   rlb       Initial Implementation                            */
/* ========================================================================= */

#include <stdio.h>
#include <cscape.h>
#include <scancode.h>
#include "colors.h"


int exit_program()
{
menu_type   menu;
sed_type    sed;
int         ret;

   menu = menu_Open();

   buildbox(menu, menu_color, 8, 28);

   menu_Printf(menu, "@p[1,4]Do you wish to exit?");
   menu_Printf(menu, "@p[4,1]@f[@[11, ]Yes@[12, ]]",
     NULL, &menu_funcs);
   menu_Printf(menu, "@p[5,1]@f[@[11, ]No@[13, ]]",
     NULL, &menu_funcs);

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, menu_color, menu_color, field_color);
   sed_SetPosition(sed, 8, 25);

   sed_Push(sed);
   sed_Repaint(sed);

   do ret = sed_Go(sed);
   while (ret > sed_GetFieldCount(sed));

   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);
   return(ret);
}
```

```
/* ========================================================================== */
/*                                  FINDFLS.C                                 */
/* ========================================================================== */
/*  This file contains the code for obtaining the list of study files in      */
/*  the current working directory.                                            */
/* ========================================================================== */
/*                             Created under Contract                         */
/*                                       to                                   */
/*                               EPA/AEERL/GECD/EMB                           */
/*                                                                            */
/*                                       by                                   */
/*                                                                            */
/*                               Radian Corporation                           */
/*                            Research Triangle Park, NC                      */
/* ========================================================================== */
/* ========================================================================== */
/*                             Revision History                               */
/* ========================================================================== */
/*    07/03/90    rlb        Initial Implementation                           */
/* ========================================================================== */

#include <stdio.h>
#include <dos.h>
#include <malloc.h>
#include "landfill.h"

unsigned compile_file_list(buffer)
char (**buffer)[13];
{
extern struct system_parms config;
struct find_t file_info;
unsigned count = 0, index = 0;
char path[256];

/* ========================================================================== */
/*                   Piece together the path + filename.                      */
/* ========================================================================== */

   strcpy(path, config.working_dir);
   strcat(path, "\\");

   strcat(path, "*.PRM");

/* ========================================================================== */
/*   Get a count of the number of files present in the working directory      */
/*   which match the file specification.  This will be used to malloc         */
/*   space for the list.                                                      */
/* ========================================================================== */

   if (! _dos_findfirst(path, _A_NORMAL, &file_info)) count++;
   while (! _dos_findnext(&file_info)) count++;

/* ========================================================================== */
/*    If there are no files, no need to go any further.  Return a NULL        */
/*    pointer and a zero count.                                               */
/* ========================================================================== */

   if (! count)
      {
      *buffer = NULL;
      return(0);
      }

/* ========================================================================== */
/*    Having gotten the file count, make space available for the list.        */
/* ========================================================================== */
```



```
   *buffer = (char (*)[13]) calloc((size_t) count, 13 * sizeof(char));

   if (*buffer == NULL)
      show_mem_error("compile_file_list", "buffer");

/* ========================================================================== */
/*                Now that we have space, fill the memory.                    */
/* ========================================================================== */

   if (! _dos_findfirst(path, _A_NORMAL, &file_info))
      strcpy((*buffer)[index++], file_info.name);

   while (! _dos_findnext(&file_info))
      strcpy((*buffer)[index++], file_info.name);

/* ========================================================================== */
/*          Now, the list has been compiled.  Return the count.               */
/* ========================================================================== */

   return(count);
}
```

```
/* ========================================================================== */
/*                                  FLSELECT.C                                */
/* ========================================================================== */
/*  This file contains the code for displaying the file selector menu.        */
/* ========================================================================== */
/*                             Created under Contract                         */
/*                                       to                                   */
/*                               EPA/AEERL/GECD/EMB                           */
/*                                                                            */
/*                                       by                                   */
/*                                                                            */
/*                                 Radian Corporation                         */
/*                             Research Triangle Park, NC                     */
/* ========================================================================== */
/* ========================================================================== */
/*                                 Revision History                           */
/* ========================================================================== */
/*    07/03/90    rlb        Initial Implementation                           */
/* ========================================================================== */

#include <stdio.h>
#include <cscape.h>
#include <scancode.h>
#include <string.h>
#include "landfill.h"
#include "helpasgn.h"
#include "colors.h"

#define UP_ARROW      24
#define DOWN_ARROW    25
#define SPACE         32

int file_selector(list, count)
char (*list)[13];
unsigned count;
{
menu_type   menu, list_menu, key_menu;
sed_type    sed, list_sed, key_sed;
int         i, ret, right_pad, curr_offset = 0, old_offset, menu_height;
int down_indicator = SPACE, up_indicator = SPACE;
char format[80], *pointer;

   if (count > 7) down_indicator = DOWN_ARROW;

   menu = menu_Open();

   menu_Printf(menu, "@p[0,0]@c[%c]+@[24,-]+", menu_color);
   menu_Printf(menu, "@p[1,0]| Choose a Study to Edit |");
   menu_Printf(menu, "@p[2,0]|--@[22,-]|");
   menu_Printf(menu, "@p[3,0]| |");
   menu_Printf(menu, "@p[3,25]|");
   menu_Printf(menu, "@p[4,0]| |");
   menu_Printf(menu, "@p[4,25]|");
   menu_Printf(menu, "@p[5,0]| |");
   menu_Printf(menu, "@p[5,25]|");
   menu_Printf(menu, "@p[6,0]| |");
   menu_Printf(menu, "@p[6,25]|");
   menu_Printf(menu, "@p[7,0]| |");
   menu_Printf(menu, "@p[7,25]|");
   menu_Printf(menu, "@p[8,0]| |");
   menu_Printf(menu, "@p[8,25]|");
   menu_Printf(menu, "@p[9,0]| |");
   menu_Printf(menu, "@p[9,25]|");
   menu_Printf(menu, "@p[10,0]| |");
   menu_Printf(menu, "@p[10,25]|");
```



```
   menu_Printf(menu, "@p[11,0]+--@[22,-]+");

   menu_Printf(menu, "@p[3,1]@fp[#]", &down_indicator, &char_funcs);
   menu_Printf(menu, "@p[10,1]@fp[#]", &up_indicator, &char_funcs);

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, menu_color, menu_color, menu_color);
   sed_SetPosition(sed, 6, 26);

   list_menu = menu_Open();

   for (i = 0 ; i < 8 ; i++)
      menu_Printf(list_menu, "@p[%d,0]@c[%c]@[37, ]", i, menu_color);

   for (i = 0 ; i < count ; i++)
      {
      pointer = strstr(list[i], ".");
      *pointer = '\0';

      right_pad = 22 - strlen(list[i]);

      sprintf(format, "@p[%d,0]@c[%c]@f[ %s@[%d, ]]", i, menu_color,
                        list[i], (right_pad >= 0 ? right_pad : 0));

      menu_Printf(list_menu, format, NULL, &menu_funcs);
      }

   menu_Flush(list_menu);

   list_sed = sed_Open(list_menu);
   sed_SetColors(list_sed, menu_color, menu_color, field_color);
   sed_SetPosition(list_sed, 9, 29);

   sed_SetWidth(list_sed, 22);
   sed_SetHeight(list_sed, 8);

   menu_height = sed_GetMenuHeight(list_sed);

   sed_Push(sed);
   sed_Repaint(sed);
   show_key_help(&key_menu, &key_sed, generic_menu_help_string);

   sed_Repaint(list_sed);

   sed_SetActive(list_sed, TRUE);

   sed_GotoFirstField(list_sed);

   sed_DoFieldSenter(list_sed, sed_GetFieldNo(list_sed));
   sed_DoFieldFenter(list_sed, sed_GetFieldNo(list_sed));

   do
      {
      sed_DoFieldFkey(list_sed, sed_GetFieldNo(list_sed));

      ret = sed_GetBaton(list_sed);

      switch (ret)
         {
         case FN1:
            compile_help(FILE_SELECTOR);
            sed_SetBaton(list_sed, -1);
            break;
         }
```

```
      curr_offset = sed_GetYoffset(list_sed);

      if (curr_offset) up_indicator = UP_ARROW;
      else up_indicator = SPACE;

      if ((curr_offset + 8) < menu_height) down_indicator = DOWN_ARROW;
      else down_indicator = SPACE;

      if (curr_offset != old_offset)
         {
         sed_RepaintFields(sed);
         old_offset = curr_offset;
         }
      }
   while ((ret > sed_GetFieldCount(list_sed)) || (ret == -1));

   destroy_key_help(key_menu, key_sed);
   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);

   menu_Close(list_menu);
   sed_Close(list_sed);

   return(ret);
}
```

```
/* ========================================================================== */
/*                                  GETOUT.C                                  */
/* ========================================================================== */
/*  This file contains the code for displaying the output filename entry      */
/*  menu.                                                                     */
/* ========================================================================== */
/*                            Created under Contract                          */
/*                                      to                                    */
/*                              EPA/AEERL/GECD/EMB                            */
/*                                                                            */
/*                                      by                                    */
/*                                                                            */
/*                              Radian Corporation                            */
/*                           Research Triangle Park, NC                       */
/* ========================================================================== */
/* ========================================================================== */
/*                              Revision History                              */
/* ========================================================================== */
/*    07/03/90    rlb       Initial Implementation                            */
/* ========================================================================== */

#include <stdio.h>
#include <cscape.h>
#include <scancode.h>
#include "landuser.h"
#include "helpasgn.h"
#include "colors.h"

int get_output_filename(filename)
char *filename;
{
menu_type   menu, key_menu;
sed_type    sed, key_sed;
int         ret;

   reset_insert_status();

   menu = menu_Open();

   menu_Printf(menu, "@p[0,0]@c[%c]+@[35,-]+", menu_color);
   menu_Printf(menu, "@p[1,0]|");
   menu_Printf(menu, "@p[1,8]Enter Output Filename");
   menu_Printf(menu, "@p[1,36]|");
   menu_Printf(menu, "@p[2,0]|@[35,-]|");
   menu_Printf(menu, "@p[3,0]|");
   menu_Printf(menu, "@p[3,36]|");
   menu_Printf(menu, "@p[4,0]|");
   menu_Printf(menu, "@p[4,36]|");
   menu_Printf(menu, "@p[5,0]|");
   menu_Printf(menu, "@p[5,36]|");
   menu_Printf(menu, "@p[6,0]+@[35,-]+");
   menu_Printf(menu, "@p[4,13]@f[@[12,#]]",
     filename, &dosfile_funcs);

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, menu_color, menu_color, field_color);
   sed_SetPosition(sed, 9, 21);

   sed_Push(sed);
   sed_Repaint(sed);
   show_key_help(&key_menu, &key_sed, studname_help_string);

   do
      {
```



```
      ret = sed_Go(sed);

      if (ret == FN1)
         {
         compile_help(GET_OUTPUT_FILENAME);
         sed_SetBaton(sed, -1);
         }
      }
   while ((ret > sed_GetFieldCount(sed)) ||
          ((strlen(filename) == 0) && (ret)));

   destroy_key_help(key_menu, key_sed);
   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);
   return(ret);
}
```

```
/* ========================================================================== */
/*                                 HELP_SCR.C                                 */
/* ========================================================================== */
/*  This file contains the code for displaying help information for a         */
/*  specific menu.                                                            */
/* ========================================================================== */
/*                             Created under Contract                         */
/*                                       to                                   */
/*                               EPA/AEERL/GECD/EMB                           */
/*                                                                            */
/*                                       by                                   */
/*                                                                            */
/*                               Radian Corporation                           */
/*                            Research Triangle Park, NC                      */
/* ========================================================================== */
/* ========================================================================== */
/*                             Revision History                               */
/* ========================================================================== */
/*    07/03/90   rlb       Initial Implementation                             */
/* ========================================================================== */

#include <stdio.h>
#include <cscape.h>
#include <scancode.h>
#include <string.h>
#include <time.h>
#include "landfill.h"
#include "landuser.h"
#include "helpasgn.h"
#include "colors.h"

#define DELAY 2 /* Delay time for help warning messages in seconds */

/* ========================================================================== */
/*      IBM specific values for character codes for various symbols.          */
/* ========================================================================== */

#define UP_ARROW      24
#define DOWN_ARROW    25
#define SPACE         32

/* ========================================================================== */
/*  This function positions the file pointer at the appropriate location     */
/* in help file and passes control to the function that actually displays     */
/* the information.                                                           */
/* ========================================================================== */

int compile_help(sed_id)
int sed_id;
{
char buffer[256], filename[256];
int current_sed;
char *help_lines, *help_level;
FILE *help_file;
extern struct system_parms config;

void display_no_help(), display_missing_help();

/* ========================================================================== */
/*  The help file is expected to be in the directory containing the           */
/*  program.  If it is not found, display the help not found popup.           */
/* ========================================================================== */

   strcpy(filename, config.landfill_dir);
   strcat(filename, "LANDFILL.HLP");
```



```
   help_file = fopen(filename, "r");

   if (help_file == NULL)
      {
      display_no_help();

      return(0);
      }

/* ========================================================================== */
/*  Start from the beginning and search until the symbol '@@' is found.       */
/*  This signals the beginning of help information and the number that        */
/*  follows indicates which help entry it is.  Look through the file         */
/*  until the correct entry is found.  If it is not, display the missing      */
/*  help popup.                                                               */
/* ========================================================================== */

   while (fgets(buffer, 255, help_file) != NULL)
      {
      help_level = strstr(buffer, "@@");
      if (help_level != NULL)
         {
         sscanf(help_level + 2, "%d", &current_sed);
         if (current_sed == sed_id) break;
         }
      }

   if (current_sed != sed_id)
      {
      display_missing_help(sed_id);

      return(1);
      }

/* ========================================================================== */
/*      Ok, the file is at the right place.  Build the help screen.           */
/* ========================================================================== */

   show_help_screen(help_file);

   return(0);
}



int show_help_screen(help_file)
FILE *help_file;
{
unsigned old_cursor;
menu_type   menu, view_menu, key_menu;
sed_type    sed, view_sed, key_sed;
int       ret, line = 0, left_space, right_space;
char up_indicator = SPACE, down_indicator = SPACE, string[255];
int curr_offset = 0 , old_offset = 0;

/* ========================================================================== */
/*                              Build the view box                            */
/* ========================================================================== */

   old_cursor = vid_GetCursorType();
   vid_SetCursorType(CURSOR_NONE);

   menu = menu_Open();

   build_scrollbox(menu, info_color, 24, 80);
```

```
   menu_Printf(menu, "@p[3,1]@f[#]",
     &down_indicator, &char_funcs);
   menu_Printf(menu, "@p[22,1]@f[#]",
     &up_indicator, &char_funcs);


   if (fgets(string, 255, help_file) != NULL)
      {
      string[strlen(string) - 1] = '\0';
      left_space = (int) (78 - strlen(string)) / 2 + 1;
      right_space = 78 - left_space - strlen(string);
      menu_Printf(menu, "@p[1,1]@c[\x70]@[%d, ]%s@[%d, ]@c[\x07]",
         left_space, string, right_space);
      }
   else
      {
      menu_Close(menu);
      return(0);
      }

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, info_color, info_color, info_color);
   sed_SetPosition(sed, 0, 0);

   sed_Push(sed);
   sed_Repaint(sed);
   show_key_help(&key_menu, &key_sed, tabular_report_key_help);

/* ========================================================================= */
/*                  Build the information to be viewed.                      */
/* ========================================================================= */

   view_menu = menu_Open();

   while (fgets(string, 255, help_file) != NULL)
      {
/* ========================================================================= */
/*  Keep searching until the beginning of the next help block is found or    */
/*  the end of file is reached.                                              */
/* ========================================================================= */

      if (strstr(string, "@@") == NULL)
         menu_Printf(view_menu, "@c[%c]%s", info_color, string);
      else
         break;
      }

   menu_Flush(view_menu);

   view_sed = sed_Open(view_menu);
   sed_SetColors(view_sed, info_color, info_color, info_color);
   sed_SetPosition(view_sed, 3, 3);

   sed_SetWidth(view_sed, 76);
   sed_SetHeight(view_sed, 20);

   line = sed_GetMenuHeight(view_sed);

   if (line > 20)
      {
      down_indicator = DOWN_ARROW;
      sed_RepaintFields(sed);
      }
```



```
   sed_Repaint(view_sed);

   do
      {
      ret = kb_Read();

      switch (ret)
         {
         case PGUP:
            sed_ScrollUp(view_sed, 18);
            break;

         case PGDN:
            sed_ScrollDown(view_sed, 18);
            break;

         case UP:
            sed_ScrollUp(view_sed, 1);
            break;

         case DOWN:
            sed_ScrollDown(view_sed, 1);
            break;
         }

      curr_offset = sed_GetYoffset(view_sed);

      if (curr_offset) up_indicator = UP_ARROW;
      else up_indicator = SPACE;

      if ((line - curr_offset) > 20) down_indicator = DOWN_ARROW;
      else down_indicator = SPACE;

      if (curr_offset != old_offset)
         {
         sed_RepaintFields(sed);
         old_offset = curr_offset;
         }
      }
   while (ret != ESC);

   destroy_key_help(key_menu, key_sed);
   sed_Pop(sed);

   menu_Close(view_menu);
   menu_Close(menu);

   sed_Close(view_sed);
   sed_Close(sed);

   vid_SetCursorType(old_cursor);

   return(ret);
}



void display_no_help()
{
menu_type   menu;
sed_type    sed;
time_t   org_count, curr_count;

   menu = menu_Open();
```

```
   menu_Printf(menu, "@p[0,0]@c[\x07]+@[32,-]+");
   menu_Printf(menu, "@p[1,0]¦");
   menu_Printf(menu, "@p[1,33]¦");
   menu_Printf(menu, "@p[2,0]¦     Help File Not Present");
   menu_Printf(menu, "@p[2,33]¦");
   menu_Printf(menu, "@p[3,0]¦");
   menu_Printf(menu, "@p[3,33]¦");
   menu_Printf(menu, "@p[4,0]+@[32,-]+");

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, '\x07', '\x07', '\x70');
   sed_SetPosition(sed, 10, 23);

   sed_Push(sed);
   sed_Repaint(sed);

   time(&org_count);
   while (time(&curr_count) < org_count + DELAY);

   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);
   return;
}



void display_missing_help(number)
int number;
{
menu_type   menu;
sed_type   sed;
long       i;
time_t   org_count, curr_count;

   menu = menu_Open();

   menu_Printf(menu, "@p[0,0]+@[32,-]+");
   menu_Printf(menu, "@p[1,0]¦");
   menu_Printf(menu, "@p[1,33]¦");
   menu_Printf(menu, "@p[2,0]¦ No Help Available for Item     ¦");
   menu_Printf(menu, "@p[3,0]¦");
   menu_Printf(menu, "@p[3,33]¦");
   menu_Printf(menu, "@p[4,0]+@[32,-]+");
   menu_Printf(menu, "@p[2,29]@fp[###]", &number, &susint_funcs);

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, '\x07', '\x07', '\x70');
   sed_SetPosition(sed, 10, 23);

   sed_Push(sed);
   sed_Repaint(sed);

   time(&org_count);
   while (time(&curr_count) < org_count + DELAY);

   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);
   return;
```



```
}
```

```
/* ========================================================================== */
/*                                  KCALCMEN.C                                */
/* ========================================================================== */
/*  This file contains the code for displaying the entry screen of            */
/*  parameters necessary to calculate a k value.                              */
/* ========================================================================== */
/*                            Created under Contract                          */
/*                                    to                                      */
/*                              EPA/AEERL/GECD/EMB                            */
/*                                                                            */
/*                                    by                                      */
/*                                                                            */
/*                              Radian Corporation                            */
/*                           Research Triangle Park, NC                       */
/* ========================================================================== */
/* ========================================================================== */
/*                              Revision History                              */
/* ========================================================================== */
/*    07/03/90   rlb       Initial Implementation                             */
/* ========================================================================== */

#include <stdio.h>
#include <ctype.h>
#include <cscape.h>
#include <scancode.h>
#include "landuser.h"
#include "landfill.h"
#include "defaults.h"
#include "helpasgn.h"
#include "colors.h"

int calculate_k_menu(k_calc_info, model_info)
struct k_calc_parms *k_calc_info;
struct model_parms *model_info;
{
void (*old_fkey)();
boolean check_nonzero(struct k_calc_parms *, double);
menu_type   menu, key_menu;
sed_type    sed, key_sed;
int         ret;
double new_k;

void recalc_fkey();
double calculate_k(struct k_calc_parms *, struct model_parms *);

/* ========================================================================== */
/*  To implement auto recalc, I need to know when each value is change.       */
/*  Consequently, the normal function key function is swapped for a new       */
/*  one.                                                                      */
/* ========================================================================== */

   old_fkey = susdbl_funcs.fkey;
   susdbl_funcs.fkey = recalc_fkey;

   reset_insert_status();

   menu = menu_Open();

   buildbox(menu, menu_color, 21, 60);

   menu_Printf(menu, "@p[1,10]Calculate Methane Gas Generation Constant");
   menu_Printf(menu, "@p[4,2]Average Well Depth");
   menu_Printf(menu, "@p[4,50]ft");
   menu_Printf(menu, "@p[5,2]Average Stabilized");
   menu_Printf(menu, "@p[6,7]Radius of Influence");
   menu_Printf(menu, "@p[6,50]ft");
```



```
   menu_Printf(menu, "@p[7,2]Refuse Density");
   menu_Printf(menu, "@p[7,50]Mg/ft^3");
   menu_Printf(menu, "@p[8,2]Fraction of");
   menu_Printf(menu, "@p[9,7]Decomposable Refuse");
   menu_Printf(menu, "@p[10,2]Methane Generation");
   menu_Printf(menu, "@p[11,7]Potential (Lo)");
   menu_Printf(menu, "@p[11,50]ft^3/Mg");
   menu_Printf(menu, "@p[12,2]Average Stabilized");
   menu_Printf(menu, "@p[13,7]Flow Rate per Well");
   menu_Printf(menu, "@p[13,50]ft^3/min");
   menu_Printf(menu, "@p[14,2]Average Age of Refuse");
   menu_Printf(menu, "@p[14,50]yr");
   menu_Printf(menu, "@p[16,0]|@[58,-]|");
   menu_Printf(menu, "@p[18,5]Calculated Value of k");
   menu_Printf(menu, "@p[18,47]yr^-1");
   menu_Printf(menu, "@p[4,32]@f[@[17,#]]",
     &(k_calc_info->well_depth), &susdbl_funcs);
   menu_Printf(menu, "@p[6,32]@f[@[17,#]]",
     &(k_calc_info->radius), &susdbl_funcs);
   menu_Printf(menu, "@p[7,32]@f[@[17,#]]",
     &(k_calc_info->density), &susdbl_funcs);
   menu_Printf(menu, "@p[9,32]@f[@[17,#]]",
     &(k_calc_info->fraction), &susdbl_funcs);
   menu_Printf(menu, "@p[11,32]@f[@[17,#]]",
     &(model_info->l0), &susdbl_funcs);
   menu_Printf(menu, "@p[13,32]@f[@[17,#]]",
     &(k_calc_info->flow_rate), &susdbl_funcs);
   menu_Printf(menu, "@p[14,32]@f[@[17,#]]",
     &(k_calc_info->refuse_age), &susdbl_funcs);
   menu_Printf(menu, "@p[18,29]@fp[@[18,#]]",
     &new_k, &susdbl_funcs);

   new_k = model_info->k;

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, menu_color, menu_color, field_color);
   sed_SetPosition(sed, 1, 10);

   sed_Push(sed);
   sed_Repaint(sed);
   show_key_help(&key_menu, &key_sed, no_default_help_string);

   do
      {
      ret = sed_Go(sed);

/* ========================================================================== */
/*  If the parameters entered check out OK, recalc a k value.                 */
/* ========================================================================== */

      if ((ret) && (check_nonzero(k_calc_info, model_info->l0)))
         {
         sed_DoFieldFexit(sed, sed_GetFieldNo(sed));
         new_k = calculate_k(k_calc_info, model_info);
         }

      switch (ret)
         {
         case FN1:
            compile_help(CALCULATE_K_MENU);
            sed_SetBaton(sed, -1);
            break;

         case FN2:
```

```
            switch (sed_GetFieldNo(sed))
               {
               case 0:
                  k_calc_info->well_depth = DEFAULT_WELL_DEPTH;
                  break;

               case 1:
                  k_calc_info->radius = DEFAULT_INFLUENCE_RAD;
                  break;

               case 2:
                  k_calc_info->density = DEFAULT_REFUSE_DENSITY;
                  break;

               case 3:
                  k_calc_info->fraction = DEFAULT_DECOMP_FRACTION;
                  break;

               case 4:
                  if (config.default_mode == AP_MODE)
                     model_info->l0 = DEFAULT_AP42_L0;
                  else model_info->l0 = DEFAULT_REGULATORY_L0;
                  break;

               case 5:
                  k_calc_info->flow_rate = DEFAULT_FLOW_RATE;
                  break;

               case 6:
                  k_calc_info->refuse_age = DEFAULT_REFUSE_AGE;
                  break;
               }
         }

      sed_RepaintFields(sed);
      }
   while ((ret) && (ret != FN10));

   sed_DoFieldSexit(sed, sed_GetFieldNo(sed));

   if (ret == FN10) model_info->k = new_k;

   destroy_key_help(key_menu, key_sed);
   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);

   susdbl_funcs.fkey = old_fkey;

   return(ret);
}


boolean check_nonzero(k_calc_info, l0)
struct k_calc_parms *k_calc_info;
double l0;
{
boolean ret = TRUE;

   if (k_calc_info->well_depth <= 0.0)
      ret = FALSE;

   if (k_calc_info->radius <= 0.0)
      ret = FALSE;
```



```
   if (k_calc_info->density <= 0.0)
      ret = FALSE;

   if (k_calc_info->fraction <= 0.0)
      ret = FALSE;

   if (k_calc_info->flow_rate <= 0.0)
      ret = FALSE;

   if (k_calc_info->refuse_age <= 0.0)
      ret = FALSE;

   if (l0 <= 0.0)
      ret = FALSE;

   return(ret);
}


void recalc_fkey(sed)
sed_type sed;
{
int scancode, key;

   scancode = kb_Read();

   if (special_key(sed, scancode)) return;
   if (recalc_inter_field(sed, scancode)) return;
   if (inter_page(sed, scancode))  return;

   if ((ascii(scancode)) && (sed_GetRecordPos(sed) == 0) && (! kb_Insert()))
      {
        sed_SetRecord(sed, "\0", sed_GetFieldNo(sed));
      sed_UpdateCurrField(sed);
      }

   switch(scancode)
      {
      case RIGHT:
         sed_IncChar(sed);
         break;

      case LEFT:
         sed_DecChar(sed);
        break;

      case HOME:
         sed_GoHome(sed);
               break;

      case END:
         sed_GoEnd(sed);
               break;

      case INS:
         if (kb_Insert())
            vid_SetCursorType(CURSOR_HALF);
         else
            vid_SetCursorType(CURSOR_NORMAL);
         break;

      case BACKSPACE:
         if (sed_DecChar(sed)) sed_PullRight(sed);
         break;

      case DEL:
```

```
         sed_PullRight(sed);
         break;

      default:
         key = ascii(scancode);
         if (isdigit(key) || key == '-' || key == '.' || toupper(key) == 'E')
            {
            if (kb_Insert())
               sed_PushRight(sed, key);
            else
               sed_Overwrite(sed, key);

            sed_IncChar(sed);
            }

         break;
      }
}


boolean recalc_inter_field(sed, scancode)
sed_type sed;
int scancode;
{
   switch (scancode)
      {
      case ESC:
         sed_SetBaton(sed, 0);
         sed_ToggleExit(sed);
         return(TRUE);

      case ENTER:
         sed_SetBaton(sed, sed_GetFieldNo(sed)+1);
         sed_IncField(sed);
         sed_ToggleExit(sed);
         return(TRUE);

      case UP:
         sed_DecField(sed);
         sed_SetBaton(sed, sed_GetFieldNo(sed)+1);
         sed_ToggleExit(sed);
         return(TRUE);

      case DOWN:
         sed_IncField(sed);
         sed_SetBaton(sed, sed_GetFieldNo(sed)+1);
         sed_ToggleExit(sed);
         return(TRUE);

      default:
         break;
      }

   return(FALSE);
}
```

```
/* ========================================================================== */
/*                                  KCALCMN.C                                 */
/* ========================================================================== */
/*  This file contains the code for calculating a k value from input          */
/*  parameters obtained by testing.                                           */
/* ========================================================================== */
/*                             Created under Contract                         */
/*                                       to                                   */
/*                               EPA/AEERL/GECD/EMB                           */
/*                                                                            */
/*                                       by                                   */
/*                                                                            */
/*                               Radian Corporation                           */
/*                            Research Triangle Park, NC                      */
/* ========================================================================== */
/* ========================================================================== */
/*                             Revision History                               */
/* ========================================================================== */
/*    07/03/90    rlb        Initial Implementation                           */
/* ========================================================================== */

#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include "landfill.h"

#include "constant.h"

/* ========================================================================== */
/*           Determine the calculation limitations of the system.             */
/* ========================================================================== */

#define FINE_ITERATIONS 10000
#define COARSE_ITERATIONS 50
#define TOLERANCE 0.000001

/* ========================================================================== */
/*  This function bears a little description.  The function cannot be         */
/*  solved analytically for k.  Rather, for a given set of parameters,        */
/*  k must be approximated.                                                   */
/*                                                                            */
/*  The function can be shown to have two roots.  One root lies in the        */
/*  interval 0 and 1/refuse_age (the maximum of the function) and the         */
/*  other between 1/refuse_age and infinity.  The root of interest lies       */
/*  closest to the origin.  Therefore, in order to insure the correct         */
/*  root is found, this function implements two different approximation       */
/*  methods for the root:  BISECTION (used to adjust the approximation        */
/*  corsely) and NEWTON-RAPHSON (used to make fine adjustments to the         */
/*  approximation.                                                            */
/*                                                                            */
/*  BISECTION attempts to find the root on an interval by adding or           */
/*  subtracting ever decreasing corrections to a guess lying on the           */
/*  interval in an attempt to zero in on the root.                            */
/*                                                                            */
/*  After a rough but close estimation of the root, NEWTON-RAPHSON makes      */
/*  changes to the estimation be altering the value based on the              */
/*  derivative of the function.  Given a value close to the root, this        */
/*  method requires only a few iterations to achieve acceptable accuracy.    */
/* ========================================================================== */

double calculate_k(k_calc_info, model_info)
struct k_calc_parms *k_calc_info;
struct model_parms *model_info;
{
double affected_depth, refuse_volume, affected_refuse_mass;
double adjusted_10, old_k, new_k, delta_k, correction;
```



```
double old_res, new_res;
double max_value;

int count = 0;

double landfill_function(), landfill_derivative();

/* ========================================================================== */
/*  The affected depth cannot exceed the landfill depth.  This assumes        */
/*  the well depth is 0.75 of the landfill depth in order to determine if     */
/*  this is so.                                                               */
/* ========================================================================== */

   affected_depth = k_calc_info->well_depth + k_calc_info->radius;

   if (affected_depth > k_calc_info->well_depth / 0.75)
      affected_depth = k_calc_info->well_depth / 0.75;

/* ========================================================================== */
/*            Calculate parameters required for the equation.                 */
/* ========================================================================== */

   refuse_volume = PI * affected_depth * k_calc_info->radius *
                        k_calc_info->radius;

   affected_refuse_mass = refuse_volume * k_calc_info->density;

   adjusted_10 = k_calc_info->fraction * model_info->10;

/* ========================================================================== */
/*  If the landfill function is less than zero at the local maximum           */
/*  (which can be shown to be 1/refuse_age), no solution can be found.        */
/*  Automatically return a -99999.                                            */
/* ========================================================================== */

  if ((max_value = landfill_function((1 / k_calc_info->refuse_age),
                        k_calc_info->refuse_age,
                        k_calc_info->flow_rate, adjusted_10,
                        affected_refuse_mass)) < 0) return(-99999);

/* ========================================================================== */
/*  Just in case the parameters are already such that the maximum is the      */
/*  zero of the function, check to see and return it.                         */
/* ========================================================================== */


   if (max_value < TOLERANCE) return(max_value);

/* ========================================================================== */
/*  Set up for the BISECTION algorithm.  The initial correction value is      */
/*  set to 1/refuse_age (the maximum of the function).  Two values for k      */
/*  are estimated.  For the zero case, the landfill function will return      */
/*  its most negative value (on the interval of interest).  The direction     */
/*  of search is set initially to increasing k value.                         */
/* ========================================================================== */

   delta_k = 1 / (2 * k_calc_info->refuse_age);

   old_k = 0.0;

   old_res = landfill_function(old_k, k_calc_info->refuse_age,
                        k_calc_info->flow_rate, adjusted_10,
                           affected_refuse_mass);

   old_k = delta_k;
```

```
   while (count++ < COARSE_ITERATIONS)
      [
/* ========================================================================= */
/*      Determine the function value using the corrected value of k.        */
/* ========================================================================= */

      new_res = landfill_function(old_k, k_calc_info->refuse_age,
                        k_calc_info->flow_rate, adjusted_l0,
                           affected_refuse_mass);

/* ========================================================================= */
/*  If we manage to determine the k value accurately enough here, return     */
/*  the value and ignore the rest of the function.                           */
/* ========================================================================= */

      if (fabs(new_res) < TOLERANCE) return(old_k);

/* ========================================================================= */
/*  As soon as the new value of the landfill function changes sign from      */
/*  old value, reverse the correction direction and half the correction      */
/*  factor.                                                                  */
/* ========================================================================= */

      if (old_res * new_res < 0.0) delta_k = -(delta_k) / 2.0;

/* ========================================================================= */
/*  Correct the old value of k and set the new result to the old result      */
/* ========================================================================= */

      old_k += delta_k;
      old_res = new_res;
      }

/* ========================================================================= */
/*  Here begins the NEWTON-RAPHSON step.  Because BISECTION is slow, we      */
/*  can't wait until a root is found with an acceptable tolerance.  Use      */
/*  this portion to refine the root.                                         */
/* ========================================================================= */

   while (count++ < FINE_ITERATIONS)
      {
      correction = landfill_function(old_k, k_calc_info->refuse_age,
                        k_calc_info->flow_rate, adjusted_l0,
                           affected_refuse_mass);

      correction /= landfill_derivative(old_k, k_calc_info->refuse_age);

      new_k = old_k - correction;

      if ((old_k - new_k) < TOLERANCE) return(new_k);

      old_k = new_k;
      ]

/* ========================================================================= */
/*       Returns a -99999 if no value was found after both methods.          */
/* ========================================================================= */

   return(-99999);
}


double landfill_function(k, refuse_age, flow_rate, adjusted_l0,
                         affected_refuse_mass)
double k, refuse_age, flow_rate, adjusted_l0, affected_refuse_mass;
{
```



```
double term1, term2;

   term1 = k * exp(-k * refuse_age);

   term2 = 525960.0 * (flow_rate / (2 * adjusted_l0 * affected_refuse_mass));

   return(term1 - term2);
}


double landfill_derivative(k, refuse_age)
double k, refuse_age;
{
double term1, term2;

   term1 = exp(-k * refuse_age);

   term2 = (1 - refuse_age * k);

   return(term1 * term2);
}
```

```
/* ========================================================================== */
/*                                   KEDIT.C                                  */
/* ========================================================================== */
/*     This file contains the code for displaying the k entry screen.         */
/* ========================================================================== */
/*                             Created under Contract                         */
/*                                       to                                   */
/*                               EPA/AEERL/GECD/EMB                           */
/*                                                                            */
/*                                       by                                   */
/*                                                                            */
/*                                Radian Corporation                          */
/*                            Research Triangle Park, NC                      */
/* ========================================================================== */
/* ========================================================================== */
/*                                Revision History                            */
/* ========================================================================== */
/*    07/03/90   rlb       Initial Implementation                             */
/* ========================================================================== */

#include <stdio.h>
#include <cscape.h>
#include <scancode.h>
#include "landfill.h"
#include "landuser.h"
#include "defaults.h"
#include "helpasgn.h"
#include "colors.h"

int edit_k_value(k)
double *k;
{
menu_type   menu, key_menu;
sed_type    sed, key_sed;
int         ret;
double new_k;

   new_k = *k;

   reset_insert_status();

   menu = menu_Open();

   buildbox(menu, menu_color, 7, 42);
   menu_Printf(menu, "@p[1,5]Methane Generation Rate Constant");
   menu_Printf(menu, "@p[4,4]k :");
   menu_Printf(menu, "@p[4,31]yr^-1");
   menu_Printf(menu, "@p[4,9]@f[@[20,#]]",
     &new_k, &susdbl_funcs);

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, menu_color, menu_color, field_color);
   sed_SetPosition(sed, 9, 19);

   sed_Push(sed);
   sed_Repaint(sed);
   show_key_help(&key_menu, &key_sed, generic_help_string);

   do
      {
      ret = sed_Go(sed);

      switch (ret)
         {
```



```
         case FN1:
            compile_help(EDIT_K_VALUE);
            sed_SetBaton(sed, -1);
            break;

         case FN2:
            if (config.default_mode == AP_MODE)
               new_k = DEFAULT_AP42_K;
            else new_k = DEFAULT_REGULATORY_K;
            sed_RepaintFields(sed);
            break;
         }
      }
   while ((ret != FN10) && (ret));

   if (ret == FN10) *k = new_k;

   destroy_key_help(key_menu, key_sed);
   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);
   return(ret);
}
```

```
/* ========================================================================== */
/*                                KEYHELP.C                                   */
/* ========================================================================== */
/*  This file contains the code for displaying the key help at the bottom     */
/*  of the screen.                                                            */
/* ========================================================================== */
/*                           Created under Contract                           */
/*                                    to                                      */
/*                             EPA/AEERL/GECD/EMB                             */
/*                                                                            */
/*                                    by                                      */
/*                                                                            */
/*                             Radian Corporation                             */
/*                          Research Triangle Park, NC                        */
/* ========================================================================== */
/* ========================================================================== */
/*                           Revision History                                 */
/* ========================================================================== */
/*    07/03/90   rlb       Initial Implementation                             */
/* ========================================================================== */

#include <stdio.h>
#include <cscape.h>
#include <string.h>
#include "keyhelp.h"
#include "colors.h"

int show_key_help(menu, sed, help_string)
menu_type   *menu;
sed_type    *sed;
char *help_string;
{
char help_line[85];

   strcenter(strcpy(help_line, help_string), 80);

   *menu = menu_Open();

   menu_Printf(*menu, "@p[0,0]@c[%c]%s@c[\x07]", keyhelp_color, help_line);

   menu_Flush(*menu);

   *sed = sed_Open(*menu);
   sed_SetColors(*sed, keyhelp_color, keyhelp_color, keyhelp_color);
   sed_SetPosition(*sed, 24, 0);

   sed_Push(*sed);
   sed_Repaint(*sed);

   return(0);
}



int destroy_key_help(menu, sed)
menu_type menu;
sed_type sed;
{

   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);
   return(0);
}
```

```
/* ========================================================================== */
/*                                   MAIN.C                                   */
/* ========================================================================== */
/*  This file contains the main controlling module for the program as         */
/*  well as the functions that initialize studies.                            */
/* ========================================================================== */
/*                             Created under Contract                         */
/*                                       to                                   */
/*                               EPA/AEERL/GECD/EMB                           */
/*                                                                            */
/*                                       by                                   */
/*                                                                            */
/*                                  Radian Corporation                        */
/*                              Research Triangle Park, NC                    */
/* ========================================================================== */
/* ========================================================================== */
/*                              Revision History                              */
/* ========================================================================== */
/*    07/03/90    rlb       Initial Implementation                            */
/*    01/17/94    rlb       Changed study retrieval routine to support        */
/*                             the codisposal/no codisposal factors.          */
/*    01/18/94    rlb       Changed init_new_study include_air_toxics flag    */
/*                             to 0 to reflect default under model            */
/*                             revision.                                      */
/*    01/29/94    rlb       Changed to permit the defaults selection.         */
/* ========================================================================== */

#include <stdio.h>
#include <ctype.h>
#include <cscape.h>
#include <dos.h>
#include <bios.h>
#include <errno.h>
#include <malloc.h>
#include <memory.h>
#include <string.h>
#include <direct.h>
#include "landfill.h"
#include "defaults.h"
#include "colors.h"
#include "colordfs.h"

extern void credits(void);
extern int study_type(void);
extern boolean edit_main(struct landfill_parms *, struct model_parms *,
                         struct k_calc_parms *, boolean *);
extern int main_menu(int);
extern int get_study_name(char *, int);
extern void no_study_selected();

static char prn_table[][6] = {"LPT1", "LPT2", "COM1", "COM2"};


/* ========================================================================== */
/*                   Declare the system parameter global                      */
/* ========================================================================== */

struct system_parms config;


#define NEW_STUDY 1
#define OLD_STUDY 2

main(argc, argv)
int argc;
char *argv[];
```



```
{
char parm_string[256];
char study_name[9];
int i, choice = 1, tab_choice = 1, type = 0, old_type;
int old_cursor;
unsigned int old_row, old_col, mode, mem, code, free;
struct k_calc_parms k_calc_info;
extern struct system_parms config;

static struct landfill_parms landfill_info = {0, 0, 0, 0, 0, 0, 0, NULL};
static struct model_results results = { 0, 0, NULL, NULL, NULL, NULL, NULL,
                                        NULL };
static struct model_parms model_info = {DEFAULT_REGULATORY_L0,
                                        DEFAULT_REGULATORY_K,
                                        DEFAULT_REGULATORY_NMOC,
                                        DEFAULT_CH4, DEFAULT_CO2, 0, NULL, 0};


boolean calc_possible = FALSE, calc_done = FALSE, exit_requested = FALSE,
        study_changed = FALSE;

void init_new_study();
int init_old_study();

/* ========================================================================== */
/*  Set the memory allocation block size to the lowest possible (since I      */
/*  am allocating memory in blocks myself, I think I'm OK here).              */
/* ========================================================================== */

   _amblksiz = 1;

/* ========================================================================== */
/*  To the best of my knowledge, the CSCAPE functions do not provide a        */
/*  means of resetting the insert status bit on entry.  This gets to be       */
/*  a real pain because the cursor keeps coming up in insert mode that        */
/*  is a result of pressing the insert key external to the program.  As       */
/*  as I hate to do it, I have implemented this kludge to handle the          */
/*  problem.  This function (see MISC_UTL.C) resets the bit indicating        */
/*  insert mode by reading, altering, and writing directly the ROM BIOS       */
/*  data area.                                                                */
/* ========================================================================== */

/*   reset_insert_status(); */

   vid_GetCursorPos(&old_row, &old_col);

   vid_Clear('\x07');

   old_cursor = vid_GetCursorType();
   vid_SetCursorType(CURSOR_NONE);

   *study_name = '\0';

/* ========================================================================== */
/*  Determine the current text mode.  If the screen is mono, remap the        */
/*  global colors to black and white.                                         */
/* ========================================================================== */

   mode = vid_GetMode();
   if (mode < 2) vid_SetMode(mode + 2);

   if ((mode == 7) || (mode == 2))
      {
      menu_color = (BLACK << 4) | WHITE;
      warning_color = (BLACK << 4) | WHITE;
      error_color = (BLACK << 4) | WHITE;
```

```
      info_color = (BLACK << 4) | WHITE;
      keyhelp_color = (WHITE << 4) | BLACK;
      field_color = (WHITE << 4) | BLACK;
      }

/* ========================================================================= */
/*  Check to see if the user has enough available memory to execute this  */
/*  application.  This is done by requesting all of the memory contained  */
/*  in the system.  This will fail but the dos service will return in     */
/*  free variable the number of paragraphs (16 bytes) available.          */
/* ========================================================================= */

#ifndef NOMEMCHK

   errno = 0;
   code = _dos_allocmem((unsigned) _bios_memsize() * 64 - 1, &free);
   if (errno == ENOMEM)
      {
      if (free < 8200)
         {
         show_mem_limitation();

         vid_SetCursorType(old_cursor);
         vid_SetCursorPos(old_row, old_col);

         exit(0);
         }
      }
   else
      _dos_freemem(free);

#endif

/* ========================================================================= */
/*                            Display title screen.                          */
/* ========================================================================= */

   credits();

/* ========================================================================= */
/*             Setup system directory and printer information                */
/* ========================================================================= */

   set_system_parms(argv[0]);

/* ========================================================================= */
/* Use the just determined default mode to set the operating parameters   */
/* for the program.                                                       */
/* ========================================================================= */

   if (config.default_mode == AP_MODE)
      {
      model_info.l0 = DEFAULT_AP42_L0;
      model_info.k = DEFAULT_AP42_K;

      if (model_info.include_air_toxics == CODISPOSAL)
         model_info.nmoc = DEFAULT_AP42_NMOC_CODIS;
      else model_info.nmoc = DEFAULT_AP42_NMOC_NOCO;
      }

/* ========================================================================= */
/*                            Process Menu Entry                             */
/* ========================================================================= */

   do
      {
```



```
      choice = main_menu(choice - 1);

      switch (choice)
         {
         case 0:
            if (exit_program() == 1) exit_requested = TRUE;
            else choice = 1;
            break;

         case 1:
            old_type = type;

            if ((type) && (study_changed))
               if (display_save_menu() == 1)
                  save_study_info(study_name, landfill_info, model_info,
                                  calc_possible);

            switch(type = study_type())
               {
               case 1:
                  if (get_study_name(study_name, type))
                     {
                     init_new_study(&landfill_info, &model_info,
                                    &k_calc_info);
                     calc_possible = calc_done = FALSE;
                     study_changed = TRUE;
                     }
                  else type = old_type;

                  break;

               case 2:
                  if (get_study_name(study_name, type))
                     {
                     if (init_old_study(study_name, &landfill_info,
                                        &model_info, &k_calc_info,
                                        &calc_possible)) type = 0;
                     else study_changed = calc_done = FALSE;
                     }
                  else type = old_type;

                  break;

               default:
                  type = old_type;
               }

            break;

         case 2:
            if (type) study_changed =
                        edit_main(&landfill_info, &model_info,
                                  &k_calc_info, &calc_possible);
            else no_study_selected();
            break;

         case 3:
            if ((type) && (calc_possible))
               calculation_main(&landfill_info, &model_info, &results,
                                &calc_done);
            else if (!type) no_study_selected();
            else if (! calc_possible) warn_no_data();
            break;

         case 4:
            if ((type) && (calc_done))
```

```
            report_model_calc(&landfill_info, &model_info, &results);
         else if (! type) no_study_selected();
         else if (! calc_done) warn_no_calc();
         break;

      case 5:
         configure_system();

         if (config.default_mode == AP_MODE)
            {
            model_info.l0 = DEFAULT_AP42_L0;
            model_info.k = DEFAULT_AP42_K;

            if (model_info.include_air_toxics == CODISPOSAL)
               model_info.nmoc = DEFAULT_AP42_NMOC_CODIS;
            else model_info.nmoc = DEFAULT_AP42_NMOC_NOCO;
            }
         else
            {
            model_info.l0 = DEFAULT_REGULATORY_L0;
            model_info.k = DEFAULT_REGULATORY_K;

            model_info.nmoc = DEFAULT_REGULATORY_NMOC;
            }

         break;

      case 6:
         exit_requested = TRUE;
         break;
      }
   }
while (! exit_requested);

if ((type) && (exit_requested) && (study_changed))
   if (display_save_menu() == 1)
      save_study_info(study_name, landfill_info, model_info,
                      calc_possible);

vid_SetCursorType(old_cursor);

vid_SetCursorPos(old_row, old_col);
}



int report_model_calc(landfill, model, results)
struct model_parms *model;
struct landfill_parms *landfill;
struct model_results *results;
{
static int disp_choice = 1, tab_choice = 1;
int report_choice;
char outfile[15];

   *outfile = '\0';

   do
      {
      disp_choice = display_results_menu(disp_choice - 1);

      switch (disp_choice)
         {
         case 1:
            do
               {
```



```
            tab_choice = tabular_report_menu(tab_choice - 1);

            switch (tab_choice)
               {
               case 1:
                  if (report_choice =
                     report_selector(model->air_toxics,
                        model->number_air_toxics, NULL))
                     show_results(model, landfill, results,
                                  report_choice);

                  break;

               case 2:
                  if (report_choice = report_selector(model->air_toxics,
                        model->number_air_toxics, "ALL"))
                     {
                     if (report_choice == model->number_air_toxics + 4)
                        print_results(NULL, prn_table[config.print_device],
                                  model, landfill, results,
                                  0);
                     else
                        print_results(NULL, prn_table[config.print_device],
                                  model, landfill, results,
                                  report_choice);
                     }
                  break;

               case 3:
                  if (get_output_filename(outfile))
                     {
                     if (report_choice = report_selector(model->air_toxics,
                        model->number_air_toxics, "ALL"))
                        {
                        if (report_choice == model->number_air_toxics + 4)
                           print_results(config.working_dir, outfile,
                                    model, landfill, results,
                                    0);
                        else
                           print_results(config.working_dir, outfile,
                                    model, landfill, results,
                                    report_choice);
                        }
                     }


                  break;
               }
            }
         while ((tab_choice > 0) && (tab_choice < 4));

         if (tab_choice == 0) tab_choice = 1;
         break;

      case 2:
         if (report_choice = report_selector(model->air_toxics,
                        model->number_air_toxics, NULL))
            switch (report_choice)
               {
               case 0:
                  break;

               case 1:
                  display_onscreen_graph(results->year_list,
                                         results->annual_ch4,
                                         landfill->current_year,
```

```
                                        results->year_count,
                                        "Methane", "Cft/yr");
                     break;

                  case 2:
                     display_onscreen_graph(results->year_list,
                                        results->annual_co2,
                                        landfill->current_year,
                                        results->year_count,
                                        "Carbon Dioxide", "Cft/yr");
                     break;

                  case 3:
                     display_onscreen_graph(results->year_list,
                                        results->annual_nmoc,
                                        landfill->current_year,
                                        results->year_count,
                                        "NMOC", "Mg/yr");

                     break;

                  default:
                     display_onscreen_graph(results->year_list,
                                        results->annual_air_toxics[report
_choice - 4],
                                        landfill->current_year,
                                        results->year_count,
                                        model->air_toxics[report_choice -
 4].chemical_name,
                                        "Cft/yr");
                     break;
                  }

            break;
         }
      }
   while ((disp_choice > 0) && (disp_choice < 3));

   if (disp_choice == 0) disp_choice = 1;
}



int set_system_parms(path)
char *path;
{
FILE *config_file;
int i, len;
boolean trap = FALSE;
char string[256];

extern struct system_parms config;

/* ================================================================== */
/*  This looks kind of complicated but it's not.  The intent here is to  */
/*  determine the landfill system directory from the arguments passed by  */
/*  DOS to the program.  The first argument (0) is the full path to the  */
/*  program being executed.  This section starts at the end of the arg   */
/*  string and search backward, looking for the backslash.  For each     */
/*  character encountered before the backslash, a null is put into the   */
/*  target string.  As soon as the backslash is found, it is copied to   */
/*  the target string and all other characters to the first one are also  */
/*  placed in the string.                                                */
/* ================================================================== */

   for (i = strlen(path) - 1; i >= 0 ; i--)
```



```
      {
      if (trap)
         config.landfill_dir[i] = path[i];
      else
         {
         if (path[i] == '\\')
            {
            config.landfill_dir[i] = path[i];
            trap = TRUE;
            }
         else
            config.landfill_dir[i] = '\0';
         }
      }

   strcpy(string, config.landfill_dir);
   strcat(string, "LANDFILL.CFG");

   config_file = fopen(string, "r");

   if (config_file == NULL)
      {
      config_file = fopen(string, "w");

/* ================================================================== */
/*   Get the current working directory and place it in a place so it can  */
/*   be changed if the user wants.                                      */
/* ================================================================== */

      len = strlen(getcwd(config.working_dir, 255));

      if (*(config.working_dir + len - 1) == '\\')
         *(config.working_dir + len - 1) = '\0';

/* ================================================================== */
/*          Set default print device to 0 for PRN (1 = AUX)            */
/* ================================================================== */

      config.print_device = 0;
      config.default_mode = 0;
      config.years_passed_closure = 50;

      fprintf(config_file, "%s\n", config.working_dir);
      fprintf(config_file, "%d\n", config.print_device);
      fprintf(config_file, "%d\n", config.default_mode);
      fprintf(config_file, "%d\n", config.years_passed_closure);
      }
   else
      {
/* ================================================================== */
/*   Get the current working directory and place it in a place so it can  */
/*   be changed if the user wants.                                      */
/* ================================================================== */

      fscanf(config_file, "%s", config.working_dir);

/*       len = strlen(getcwd(config.working_dir, 255));

      if (*(config.working_dir + len - 1) == '\\')
         *(config.working_dir + len - 1) = '\0';
*/

      fscanf(config_file, "%d", &(config.print_device));
      fscanf(config_file, "%d", &(config.default_mode));
      fscanf(config_file, "%d", &(config.years_passed_closure));
      }
```

```
   fclose(config_file);

   return(0);
}



void init_new_study(landfill_info, model_info, k_calc_info)
struct landfill_parms *landfill_info;
struct model_parms *model_info;
struct k_calc_parms *k_calc_info;
{
struct dosdate_t date;
extern struct system_parms config;

void initialize_air_toxics(char *, char *, struct air_toxic_structure **,
                           int *);

   _dos_getdate(&date);

   landfill_info->year_opened = (int) (date.year - DEFAULT_YEARS_PRIOR);
   landfill_info->current_year = (int) date.year;

   landfill_info->capacity = 0.0;

   landfill_info->refuse_in_place =
      (double *) calloc(DEFAULT_YEARS_PRIOR, sizeof(double));

   if (landfill_info->refuse_in_place == NULL)
      show_mem_error("init_new_study", "refuse_in_place");

   landfill_info->number_of_periods = DEFAULT_YEARS_PRIOR;

   if (config.default_mode == REGULATORY_MODE)
      {
      model_info->l0 = DEFAULT_REGULATORY_L0;
      model_info->k = DEFAULT_REGULATORY_K;

      model_info->nmoc = DEFAULT_REGULATORY_NMOC;
      }
   else
      {
      model_info->l0 = DEFAULT_AP42_L0;
      model_info->k = DEFAULT_AP42_K;
      model_info->nmoc = DEFAULT_AP42_NMOC_NOCO;
      }

   model_info->ch4 = DEFAULT_CH4;
   model_info->co2 = DEFAULT_CO2;
   model_info->include_air_toxics = NOCODISPOSAL;

   initialize_air_toxics(config.landfill_dir, "airtoxic.txt",
                         &(model_info->air_toxics),
                         &(model_info->number_air_toxics));

   k_calc_info->well_depth = DEFAULT_WELL_DEPTH;
   k_calc_info->radius = DEFAULT_INFLUENCE_RAD;
   k_calc_info->density = DEFAULT_REFUSE_DENSITY;
   k_calc_info->fraction = DEFAULT_DECOMP_FRACTION;
   k_calc_info->flow_rate = DEFAULT_FLOW_RATE;
   k_calc_info->refuse_age = DEFAULT_REFUSE_AGE;
}


int init_old_study(study_name, landfill_info, model_info, k_calc_info,
```



```
                   calc_possible)
char *study_name;
struct landfill_parms *landfill_info;
struct model_parms *model_info;
struct k_calc_parms *k_calc_info;
boolean *calc_possible;
{
int i;
char string[256], filename[15];
extern struct system_parms config;
FILE *parameters;

void initialize_air_toxics(char *, char *, struct air_toxic_structure **,
                           int *);

   strcpy(string, config.working_dir);
   strcat(string, "\\");
   strcat(string, study_name);
   strcat(string, ".prm");

   parameters = fopen(string, "r");

   if (parameters == NULL)
      {
      show_file_error();
      return(1);
      }

   fscanf(parameters, "%lf %lf %lf %lf %lf %d",  &(model_info->l0),
               &(model_info->k), &(model_info->nmoc),
               &(model_info->ch4), &(model_info->co2),
               &(model_info->include_air_toxics));

   fscanf(parameters, "%d %d %d", &(landfill_info->year_opened),
                                  &(landfill_info->current_year),
                                  &(landfill_info->year_closed));

   fscanf(parameters, "%lf %lf %lf", &(landfill_info->capacity),
                                  &(landfill_info->avg_acceptance_rate),
                                  &(landfill_info->last_rate));

   fscanf(parameters, "%d", (int *) calc_possible);

   fscanf(parameters, "%d", &(landfill_info->number_of_periods));

   if (landfill_info->refuse_in_place != NULL)
      free(landfill_info->refuse_in_place);

   landfill_info->refuse_in_place =
      (double *) calloc((size_t) landfill_info->number_of_periods,
               sizeof(double));

   if (landfill_info->refuse_in_place == NULL)
      show_mem_error("init_new_study", "refuse_in_place");

   for (i = 0 ; i < landfill_info->number_of_periods ; i++)
      fscanf(parameters, "%lf", &(landfill_info->refuse_in_place[i]));

   fclose(parameters);

   strcpy(string, config.working_dir);
   strcat(string, "\\");

   strcpy(filename, study_name);
   strcat(filename, ".atx");
```

```
   if (model_info->include_air_toxics != NO_AT)
      {
      initialize_air_toxics(string, filename,
                            &(model_info->air_toxics),
                            &(model_info->number_air_toxics));
      }
   else
      {
      model_info->number_air_toxics = 0;
      if (model_info->air_toxics != NULL) free(model_info->air_toxics);
      model_info->air_toxics = NULL;
      }

   k_calc_info->well_depth = DEFAULT_WELL_DEPTH;
   k_calc_info->radius = DEFAULT_INFLUENCE_RAD;
   k_calc_info->density = DEFAULT_REFUSE_DENSITY;
   k_calc_info->fraction = DEFAULT_DECOMP_FRACTION;
   k_calc_info->flow_rate = DEFAULT_FLOW_RATE;
   k_calc_info->refuse_age = DEFAULT_REFUSE_AGE;

   return(0);
}


void initialize_air_toxics(path, file, list, number)
char *path, *file;
struct air_toxic_structure **list;
int *number;
{
FILE *toxics;
int i = 0;
char string[80], filename[256];

   strcpy(filename, path);
   strcat(filename, file);

   toxics = fopen(filename, "r");

   if (toxics == NULL)
      {
      *list = NULL;
      *number = 0;
      }
   else
      {
      while (fgets(string, 79, toxics) != NULL)
         if ((strlen(string) > 5) && (isgraph(*string))) i++;

      rewind(toxics);

      *number = i;

      if (*list != NULL) free(*list);

      *list = (struct air_toxic_structure *)
         calloc((size_t) *number, sizeof(struct air_toxic_structure));

      if (*list == NULL) show_mem_error("init_new_study", "toxic_list");

      i = 0;

      while (fgets(string, 79, toxics) != NULL)
         if ((strlen(string) > 5) && (isgraph(*string)))
            {
            strcpy((*list + i)->chemical_name, midstr(string, 0, 27));
```



```
            sscanf(string + 29, "%lf %lf %lf", &((*list + i)->molecular_wt),
                        &((*list + i)->concentration[NOCODISPOSAL]),
                        &((*list + i)->concentration[CODISPOSAL]));
            i++;
            }

      fclose(toxics);
      }
}
```

```
/* ========================================================================== */
/*                                 MAINMENU.C                                 */
/* ========================================================================== */
/*         This file contains the code for displaying the main menu           */
/* ========================================================================== */
/*                             Created under Contract                         */
/*                                       to                                   */
/*                               EPA/AEERL/GECD/EMB                           */
/*                                                                            */
/*                                       by                                   */
/*                                                                            */
/*                               Radian Corporation                           */
/*                            Research Triangle Park, NC                      */
/* ========================================================================== */
/* ========================================================================== */
/*                             Revision History                               */
/* ========================================================================== */
/*    07/03/90   rlb       Initial Implementation                             */
/* ========================================================================== */

#include <stdio.h>
#include <cscape.h>
#include <scancode.h>
#include "helpasgn.h"
#include "colors.h"


int main_menu(choice)
int choice;
{
menu_type   menu, key_menu;
sed_type   sed, key_sed;
int       ret, old_cursor;

void show_main_help();

   old_cursor = vid_GetCursorType();
   vid_SetCursorType(CURSOR_NONE);

   menu = menu_Open();

   buildbox(menu, menu_color, 13, 29);
   menu_Printf(menu, "@p[1,10]Main Menu");
   menu_Printf(menu, "@p[4,1]@f[@[7, ]Specify Study@[7, ]]",
     NULL, &menu_funcs);
   menu_Printf(menu, "@p[5,1]@f[      Edit Study Data      ]",
     NULL, &menu_funcs);
   menu_Printf(menu, "@p[6,1]@f[  Calculate Air Emissions  ]",
     NULL, &menu_funcs);
   menu_Printf(menu, "@p[7,1]@f[      Display Results      ]",
     NULL, &menu_funcs);
   menu_Printf(menu, "@p[8,1]@f[     Configure Program     ]",
     NULL, &menu_funcs);
   menu_Printf(menu, "@p[10,1]@f[@[11, ]Exit@[12, ]]",
     NULL, &menu_funcs);

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, menu_color, menu_color, field_color);
   sed_SetPosition(sed, 6, 25);

   sed_Push(sed);

   sed_GotoField(sed, choice);
```



```
   sed_Repaint(sed);
   show_key_help(&key_menu, &key_sed, mainmenu_help_string);

   do
      {
      ret = sed_Go(sed);

      if (ret == FN1)
         {
         compile_help(MAIN_MENU);
         sed_SetBaton(sed, -1);
         }
      }
   while (ret > sed_GetFieldCount(sed));

   destroy_key_help(key_menu, key_sed);
   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);

   vid_SetCursorType(old_cursor);

   return(ret);
}
```

```
/* ========================================================================== */
/*                                  METHCAP.C                                 */
/* ========================================================================== */
/*  This file contains the code for displaying the L0 entry screen.           */
/* ========================================================================== */
/*                             Created under Contract                         */
/*                                       to                                   */
/*                               EPA/AEERL/GECD/EMB                           */
/*                                                                            */
/*                                       by                                   */
/*                                                                            */
/*                               Radian Corporation                           */
/*                            Research Triangle Park, NC                      */
/* ========================================================================== */
/* ========================================================================== */
/*                               Revision History                             */
/* ========================================================================== */
/*    07/03/90   rlb        Initial Implementation                            */
/* ========================================================================== */

#include <stdio.h>
#include <cscape.h>
#include <scancode.h>
#include "landfill.h"
#include "landuser.h"
#include "defaults.h"
#include "helpasgn.h"
#include "colors.h"

int l0_selection_menu(l0)
double *l0;
{
menu_type   menu, key_menu;
sed_type    sed, key_sed;
int         ret;
double    l0_scratch;

   l0_scratch = *l0;

   reset_insert_status();

   menu = menu_Open();

   buildbox(menu, menu_color, 7, 42);
   menu_Printf(menu, "@p[1,2]Methane Generation Potential of Refuse");
   menu_Printf(menu, "@p[3,36]3");
   menu_Printf(menu, "@p[4,3]Lo :");
   menu_Printf(menu, "@p[4,33]ft /Mg");

   menu_Printf(menu, "@p[4,9]@f[@[22,#]]", &l0_scratch, &susdbl_funcs);

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, menu_color, menu_color, field_color);
   sed_SetPosition(sed, 9, 18);

   sed_Push(sed);
   sed_Repaint(sed);
   show_key_help(&key_menu, &key_sed, generic_help_string);

   do
      {
      ret = sed_Go(sed);
```



```
      switch (ret)
         {
         case FN1:
            compile_help(L0_SELECTION_MENU);
            sed_SetBaton(sed, -1);
            break;

         case FN2:
            if (config.default_mode == AP_MODE)
               l0_scratch = DEFAULT_AP42_L0;
            else l0_scratch = DEFAULT_REGULATORY_L0;
            sed_RepaintFields(sed);
            break;
         }
      }
   while ((ret) && (ret != FN10));

   if (ret == FN10) *l0 = l0_scratch;

   destroy_key_help(key_menu, key_sed);
   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);
   return(ret);
}
```

```
/* ========================================================================== */
/*                                METHRATE.C                                  */
/* ========================================================================== */
/*  This file contains the code for displaying the k selection menu.          */
/* ========================================================================== */
/*                           Created under Contract                           */
/*                                   to                                       */
/*                             EPA/AEERL/GECD/EMB                             */
/*                                                                            */
/*                                   by                                       */
/*                                                                            */
/*                             Radian Corporation                             */
/*                          Research Triangle Park, NC                        */
/* ========================================================================== */
/* ========================================================================== */
/*                            Revision History                                */
/* ========================================================================== */
/*    07/03/90   rlb       Initial Implementation                             */
/*    02/08/94   rlb       Removed reference to hardwired default k.          */
/* ========================================================================== */

#include <stdio.h>
#include <cscape.h>
#include <scancode.h>
#include "helpasgn.h"
#include "colors.h"

int k_selection_menu(prev_item)
int prev_item;
{
menu_type   menu, key_menu;
sed_type    sed, key_sed;
int         ret;

   menu = menu_Open();

   buildbox(menu, menu_color, 9, 49);
   menu_Printf(menu, "@p[1,3]Edit Methane Generation Decay Rate Constant");
   menu_Printf(menu, "@p[4,1]@f[@[12, ]Use Default Value of k@[13, ]]",
     NULL, &menu_funcs);
   menu_Printf(menu, "@p[5,1]@f[@[14, ]Supply a Value of k@[14, ]]",
     NULL, &menu_funcs);
   menu_Printf(menu, "@p[6,1]@f[@[12, ]Calculate a Value of k@[13, ]]",
     NULL, &menu_funcs);

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, menu_color, menu_color, field_color);
   sed_SetPosition(sed, 8, 15);

   sed_Push(sed);
   sed_Repaint(sed);
   show_key_help(&key_menu, &key_sed, generic_menu_help_string);

   sed_GotoField(sed, prev_item);

   do
      {
      ret = sed_Go(sed);

      if (ret == FN1)
         {
         compile_help(K_SELECTION_MENU);
         sed_SetBaton(sed, -1);
         }
```



```
      }
   while ((ret > sed_GetFieldCount(sed)) && (ret != FN10));

   destroy_key_help(key_menu, key_sed);
   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);
   return(ret);
}
```

```
/* ========================================================================== */
/*                                 MISC_UTL.C                                 */
/* ========================================================================== */
/*     This file contains the code for miscellaneous support routines.        */
/* ========================================================================== */
/*                             Created under Contract                         */
/*                                       to                                   */
/*                               EPA/AEERL/GECD/EMB                           */
/*                                                                            */
/*                                       by                                   */
/*                                                                            */
/*                                 Radian Corporation                         */
/*                             Research Triangle Park, NC                     */
/* ========================================================================== */
/* ========================================================================== */
/*                                Revision History                            */
/* ========================================================================== */
/*    07/03/90    rlb        Initial Implementation                           */
/* ========================================================================== */

#include <stdio.h>
#include <string.h>
#include <dos.h>
#include <memory.h>
#include <bios.h>
#include <ctype.h>

/* ========================================================================== */
/*                This function pads with spaces to length                    */
/* ========================================================================== */

int pad_string(string, padded_length)
char *string;
int padded_length;
{
int i;

   for (i = strlen(string) ; i < padded_length ; i++)
      {
      string[i] = 32;
      }

   string[i] = 0;

   return(padded_length - strlen(string));
}

/* ========================================================================== */
/*  This function extracts a substring from a string and returns a            */
/*  a pointer to the substring memory (static).                               */
/* ========================================================================== */

char *midstr(fstr,ffrom,flen)
char *fstr;
int ffrom,flen;
{
static char wstr[256];
unsigned int wlen,newlen;

   if ((wlen = strlen(fstr)) < (ffrom + 1))
   return(0);

   strncpy(wstr, fstr + ffrom, flen);   /* copy into work storage */

   newlen = wlen - ffrom + 1;
```



```
   if (newlen <= flen)
     wstr[newlen] = 0;
   else
     wstr[flen] = 0;

   return(wstr);
}

/* ========================================================================== */
/*                    This function capitalizes a string                      */
/* ========================================================================== */

int capital_string(string)
char *string;
{
int i = 0;
char *scratch;

   scratch = string;

   while (*(scratch + i) != '\0')
      {
      *(scratch + i) = (char) toupper((int) *(scratch + i));
      i++;
      }

   return(i);
}


/* ========================================================================== */
/*  This function resets the state of the insert status bit to zero.          */
/*  This is accomplished by reading the initial state of the ROM BIOS         */
/*  keyboard status byte from the ROM BIOS data area (0040:0017H),            */
/*  clearing the most significant bit, and writing the byte back to           */
/*  memory.                                                                   */
/* ========================================================================== */


int reset_insert_status()
{
char status_byte, *pointer;

   pointer = &status_byte;

   movedata(0x40, 0x17, FP_SEG(pointer), FP_OFF(pointer), 1);
   status_byte &= 0x7F;
   movedata(FP_SEG(pointer), FP_OFF(pointer), 0x40, 0x17, 1);
}
```

```
/* ========================================================================== */
/*                                NEW-OLD.C                                   */
/* ========================================================================== */
/*  This file contains the code for displaying the study type menu.           */
/* ========================================================================== */
/*                            Created under Contract                          */
/*                                    to                                      */
/*                              EPA/AEERL/GECD/EMB                            */
/*                                                                            */
/*                                    by                                      */
/*                                                                            */
/*                              Radian Corporation                            */
/*                           Research Triangle Park, NC                       */
/* ========================================================================== */
/* ========================================================================== */
/*                              Revision History                              */
/* ========================================================================== */
/*    07/03/90   rlb       Initial Implementation                             */
/* ========================================================================== */

#include <stdio.h>
#include <cscape.h>
#include <scancode.h>
#include "helpasgn.h"
#include "colors.h"


int study_type()
{
menu_type   menu, key_menu;
sed_type    sed, key_sed;
int         ret;

   menu = menu_Open();

   buildbox(menu, menu_color, 8, 31);
   menu_Printf(menu, "@p[1,6]Specify Study Type");
   menu_Printf(menu, "@p[4,1]@f[@[10, ]New Study@[10, ]]", NULL, &menu_funcs);
   menu_Printf(menu, "@p[5,1]@f[@[10, ]Old Study@[10, ]]", NULL, &menu_funcs);

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, menu_color, menu_color, field_color);
   sed_SetPosition(sed, 8, 24);

   sed_Push(sed);
   sed_Repaint(sed);
   show_key_help(&key_menu, &key_sed, generic_menu_help_string);

   do
      {
      ret = sed_Go(sed);

      if (ret == FN1)
         {
         compile_help(STUDY_TYPE);
         sed_SetBaton(sed, -1);
         }
      }
   while (ret > sed_GetFieldCount(sed));

   destroy_key_help(key_menu, key_sed);
   sed_Pop(sed);
```



```
   menu_Close(menu);
   sed_Close(sed);
   return(ret);
}
```

```
/* ================================================================== */
/*                              PERIODS.C                             */
/* ================================================================== */
/*  This file contains functions relating to the display and verification */
/*  of the landfill operating parameters for the Landfill Air Emmisions   */
/*  model.                                                            */
/* ================================================================== */
/*                        Created under Contract                      */
/*                                to                                  */
/*                          EPA/AEERL/GECD/EMB                        */
/*                                                                    */
/*                                by                                  */
/*                                                                    */
/*                          Radian Corporation                        */
/*                       Research Triangle Park, NC                   */
/* ================================================================== */
/* ================================================================== */
/*                          Revision History                          */
/* ================================================================== */
/*    07/03/90   rlb       Initial Implementation                     */
/*    01/21/94   rlb       Changed to pass format specifier as the 3rd */
/*                           data pointer to fnsusdbl funcs           */
/* ================================================================== */

#include <math.h>
#include <stdio.h>
#include <cscape.h>
#include <scancode.h>
#include <malloc.h>
#include "landfill.h"
#include "landuser.h"
#include "defaults.h"
#include "helpasgn.h"
#include "colors.h"

/* ================================================================== */
/*      IBM specific values for character codes for various symbols.  */
/* ================================================================== */

#define UP_ARROW      24
#define DOWN_ARROW    25
#define SPACE         32


int edit_periods(landfill_info)
struct landfill_parms *landfill_info;
{
struct landfill_parms scratch_landfill_parms;
menu_type   menu, key_menu;
sed_type    sed, key_sed;
int         ret;

int copy_landfill_parms(struct landfill_parms *, struct landfill_parms *);

/* ================================================================== */
/*    First, make a copy so we don't mess things up if the user aborts. */
/* ================================================================== */

   copy_landfill_parms(&scratch_landfill_parms, landfill_info);

/* ================================================================== */
/*                  Begin defining the menu structures.               */
/*       This was done using the Look and Feel editor from Cscape     */
/* ================================================================== */

   reset_insert_status();
```



```
   menu = menu_Open();

   menu_Printf(menu, "@p[0,0]@c[%c]+@[46,-]+", menu_color);
   menu_Printf(menu, "@p[1,0]¦");
   menu_Printf(menu, "@p[1,13]Edit Refuse Data Periods");
   menu_Printf(menu, "@p[1,47]¦");
   menu_Printf(menu, "@p[2,0]¦@[22,-]-@[23,-]¦");
   menu_Printf(menu, "@p[3,0]¦ Year Opened :");
   menu_Printf(menu, "@p[3,23]¦ Current Year :");
   menu_Printf(menu, "@p[3,47]¦");
   menu_Printf(menu, "@p[4,0]¦@[22,-]-@[23,-]¦");
   menu_Printf(menu, "@p[5,0]¦");
   menu_Printf(menu, "@p[5,5]Capacity :");
   menu_Printf(menu, "@p[5,38]Mg");
   menu_Printf(menu, "@p[5,47]¦");
   menu_Printf(menu, "@p[6,0]¦@[15,-]-@[30,-]¦");
   menu_Printf(menu, "@p[7,0]¦     Year");
   menu_Printf(menu, "@p[7,16]¦     Refuse In Place (Mg)     ¦");
   menu_Printf(menu, "@p[8,0]¦--@[13,-]-@[30,-]¦");
   menu_Printf(menu, "@p[9,0]¦ ¦");
   menu_Printf(menu, "@p[9,47]¦");
   menu_Printf(menu, "@p[10,0]¦ ¦");
   menu_Printf(menu, "@p[10,47]¦");
   menu_Printf(menu, "@p[11,0]¦ ¦");
   menu_Printf(menu, "@p[11,47]¦");
   menu_Printf(menu, "@p[12,0]¦ ¦");
   menu_Printf(menu, "@p[12,47]¦");
   menu_Printf(menu, "@p[13,0]¦ ¦");
   menu_Printf(menu, "@p[13,47]¦");
   menu_Printf(menu, "@p[14,0]¦ ¦");
   menu_Printf(menu, "@p[14,47]¦");
   menu_Printf(menu, "@p[15,0]¦ ¦");
   menu_Printf(menu, "@p[15,47]¦");
   menu_Printf(menu, "@p[16,0]¦ ¦");
   menu_Printf(menu, "@p[16,47]¦");
   menu_Printf(menu, "@p[17,0]¦ ¦");
   menu_Printf(menu, "@p[17,47]¦");
   menu_Printf(menu, "@p[18,0]¦ ¦");
   menu_Printf(menu, "@p[18,47]¦");
   menu_Printf(menu, "@p[19,0]¦ ¦");
   menu_Printf(menu, "@p[19,47]¦");
   menu_Printf(menu, "@p[20,0]+--@[44,-]+");

/* ================================================================== */
/*  In this level of the menu item, the user is to edit the start and end */
/*  years and the capacity.  These parameters will be used to determine   */
/*  the range of refuse periods that can be editted.                  */
/* ================================================================== */

   menu_Printf(menu, "@p[3,16]@f[####]",
     &(scratch_landfill_parms.year_opened), &susint_funcs);
   menu_Printf(menu, "@p[3,40]@f[####]",
     &(scratch_landfill_parms.current_year), &susint_funcs);
   menu_Printf(menu, "@p[5,16]@fd3[@[21,#]]",
     &(scratch_landfill_parms.capacity), &susdbl_funcs,
     NULL, NULL, "%.0lf");

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, menu_color, menu_color, field_color);
   sed_SetPosition(sed, 2, 16);

   sed_Push(sed);
   sed_Repaint(sed);
```

```
   show_key_help(&key_menu, &key_sed, generic_help_string);

/* ========================================================================= */
/*                          Main Control Loop                                */
/*  First, the upper part of the window is activated accepting the years     */
/*  and capacity.  The data is examined to determine if it is rational       */
/*  (ie., the current year is later than the start year).  Next, the data    */
/*  is passed to the next menu so the user can specify the refuse data       */
/*  for each of the years.                                                   */
/* ========================================================================= */

   do
      {
      do
         {
         do
            {
            ret = sed_Go(sed);

            if (ret == FN1)
               [
               compile_help(EDIT_PERIODS);
               sed_SetBaton(sed, -1);
               }
            ]
         while ((ret > 3) && (ret != FN10));

         if (ret) ret = verify_landfill_data(scratch_landfill_parms);
         }
      while ((ret) && (ret != FN10));

      if (! ret) break;

      do ret = refuse_periods_menu(&scratch_landfill_parms);
      while ((ret) && (ret != FN10));
      }
   while ((ret) && (ret != FN10));

/* ========================================================================= */
/*  The user has accepted the data and since we are here, the data has       */
/*  passed verification.  Copy the data back to the original arrays.         */
/* ========================================================================= */

   if (ret == FN10)
      copy_landfill_parms(landfill_info, &scratch_landfill_parms);

   destroy_key_help(key_menu, key_sed);
   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);

   if ((ret == FN10) && (! calculate_closure_year(landfill_info)))
      display_closure_year(landfill_info);

   return(ret);
]


/* ========================================================================= */
/*       This function copies the data that compose the structures           */
/* ========================================================================= */

int copy_landfill_parms(dest, source)
struct landfill_parms *dest, *source;
|
```



```
   dest->year_opened = source->year_opened;
   dest->current_year = source->current_year;
   dest->year_closed = source->year_closed;
   dest->avg_acceptance_rate = source->avg_acceptance_rate;
   dest->last_rate = source->last_rate;
   dest->capacity = source->capacity;
   dest->refuse_in_place = source->refuse_in_place;
   dest->number_of_periods = source->number_of_periods;
}

/* ========================================================================= */
/*  This function does the value checking on the first part of the menu.     */
/* ========================================================================= */

int verify_landfill_data(data)
struct landfill_parms data;
|
   if (data.year_opened >= data.current_year)
      {
      verify_error(3);
      return(1);
      |

   if ((data.current_year - data.year_opened) > (MAX_DURATION - 1))
      {
      verify_error(6);
      return(1);
      |

   if (data.capacity <= 0.0)
      |
      verify_error(4);
      return(1);
      }

/* ========================================================================= */
/*  Because this function is called between the two parts of the menu and    */
/*  the loop will terminate on a return value of zero, this function must    */
/*  return an FN10 in order for the main loop to continue.                   */
/* ========================================================================= */

   return(FN10);
}



/* ========================================================================= */
/*  This is the second component of the Edit Periods Menu Selection. This    */
/*  item allows the user to enter in all of the necessary data for the       */
/*  the periods of operation of the landfill.                                */
/* ========================================================================= */

int refuse_periods_menu(landfill_info)
struct landfill_parms *landfill_info;
{
double *scratch_refuse;
char format[80];
char up_indicator = SPACE,  down_indicator = SPACE;
menu_type   scroll_menu, data_menu;
sed_type   scroll_sed, data_sed;
int        ret, total_years, old_offset = 0, curr_offset, i;
int *year;

/* ========================================================================= */
/*  Determine the total number of years the user wants to edit from the      */
/*  data passed in from the previous menu.  Allocate an array of year        */
```

```
/*  numbers (just for show - they are not really needed) and fill it      */
/*  with all of the year data.                                            */
/* ====================================================================== */

   total_years = landfill_info->current_year
                 - landfill_info->year_opened;

   year = (int *) calloc((size_t) total_years, sizeof(int));

   if (year == NULL)
      show_mem_error("refuse_periods_menu", "year");

   for (i =  0 ; i < total_years ; i++)
      year[i] = landfill_info->year_opened + i + 1;


/* ====================================================================== */
/*  Now, allocate enough memory to hold the refuse data points.  Copy     */
/*  all previous data into the array.                                     */
/* ====================================================================== */

   scratch_refuse = (double *) calloc((size_t) total_years, sizeof(double));

   if (scratch_refuse == NULL)
      show_mem_error("refuse_periods_menu", "scratch_refuse");

   for (i = 0; ((i < total_years) &&
      (i < landfill_info->number_of_periods)) ; i++)
      scratch_refuse[i] = landfill_info->refuse_in_place[i];

/* ====================================================================== */
/*  The menu itself is actually composed of two physical menus.  One menu */
/*  serves only to display the up and down arrows based on whether the    */
/*  actual data has been scrolled and whether more data exists but is not */
/*  displayed.  The other menu contains the actual data and is the one    */
/*  being editted.                                                        */
/* ====================================================================== */

   scroll_menu = menu_Open();

   menu_Printf(scroll_menu, "@p[0,0]@fp[#]", &up_indicator, &char_funcs);
   menu_Printf(scroll_menu, "@p[10,0]@fp[#]", &down_indicator, &char_funcs);

   menu_Flush(scroll_menu);

   scroll_sed = sed_Open(scroll_menu);
   sed_SetColors(scroll_sed, menu_color, menu_color, menu_color);
   sed_SetPosition(scroll_sed, 11, 17);

   sed_SetHeight(scroll_sed, 11);
   sed_SetWidth(scroll_sed, 1);

   data_menu = menu_Open();

   for (i = 0 ; i < total_years ; i++)
      {
      sprintf(format, "@p[%d,0]@c[%c]@fp[####]@[13, ]", i, menu_color);
      menu_Printf(data_menu, format, year + i, &susint_funcs);

      sprintf(format, "@p[%d,16]@f[@[20,#]]    ", i);
      menu_Printf(data_menu, format, scratch_refuse + i,
         &susdbl_funcs);
      }

   menu_Flush(data_menu);
```



```
/* ====================================================================== */
/*                  Define the screens and open the seds                  */
/* ====================================================================== */

   data_sed = sed_Open(data_menu);
   sed_SetColors(data_sed, menu_color, menu_color, field_color);
   sed_SetPosition(data_sed, 11, 22);

   sed_SetHeight(data_sed, 11);
   sed_SetWidth(data_sed, 40);

/* ====================================================================== */
/*  Set the initial condition of the arrows based on the number of items  */
/*  in the data menu.                                                     */
/* ====================================================================== */

   if (total_years > 11)
      down_indicator = DOWN_ARROW;
   else
      down_indicator = SPACE;

   sed_Repaint(data_sed);

   sed_Repaint(scroll_sed);
   sed_Clear(scroll_sed);
   sed_RepaintFields(scroll_sed);

   sed_SetActive(data_sed, TRUE);
   sed_SetActive(scroll_sed, TRUE);

   sed_GotoFirstField(data_sed);

   sed_DoFieldSenter(data_sed, sed_GetFieldNo(data_sed));
   sed_DoFieldFenter(data_sed, sed_GetFieldNo(data_sed));

/* ====================================================================== */
/*                               Main Loop                                */
/* Check for any special keys.  If nothing, check the condition of the   */
/* seds for scrolling and set the appropriate arrows.  (NOTE: the do loop */
/* is composed of the functions normally executed by sed_Go but, in order */
/* to trap arrow keys, etc., the function had to be broken down into its  */
/* constituent functions.)                                               */
/* ====================================================================== */

   do
      {
      do
         {
         sed_DoFieldFkey(data_sed, sed_GetFieldNo(data_sed));

         ret = sed_GetBaton(data_sed);

         switch (ret)
            {
            case FN2:
               if (warn_about_period_reset())
                  {
                  for (i = 0 ; i < total_years ; i++)
                     scratch_refuse[i] = 0.0;

                  sed_RepaintFields(data_sed);
                  }

               break;

            case FN6:
```

```
            break;

         case FN1:
            compile_help(EDIT_PERIODS);
            break;
         }

      sed_SetBaton(data_sed, -1);

      curr_offset = sed_GetYoffset(data_sed);

      if (curr_offset)
         up_indicator = UP_ARROW;
      else
         up_indicator = SPACE;

      if (total_years - curr_offset > 11)
         down_indicator = DOWN_ARROW;
      else
         down_indicator = SPACE;

      if (curr_offset != old_offset)
         {
         sed_RepaintFields(scroll_sed);
         old_offset = curr_offset;
         }
      }
   while ((ret != FN10) && (ret));

   sed_DoFieldSexit(data_sed, sed_GetFieldNo(data_sed));

/* ========================================================================== */
/*       If FN10 key was pressed, the data was accepted by the user.          */
/*      Validate the data and allow the user to correct it or abort.          */
/* ========================================================================== */

   if (ret == FN10) ret = verify_data(landfill_info->capacity,
                                      scratch_refuse, total_years);
   }
while ((ret) && (ret != FN10));

/* ========================================================================== */
/*  If we have gotten this far either the ESC or the FN10 key was pressed     */
/*  and the data passed verification.  If the FN10 key was pressed, alter     */
/*  the original array so it can hold the data and copy the scratch array     */
/*  into the new space.                                                       */
/* ========================================================================== */

if (ret == FN10)
   {
   if (landfill_info->refuse_in_place != NULL)
      free(landfill_info->refuse_in_place);

   landfill_info->refuse_in_place =
      (double *) calloc((size_t) total_years, sizeof(double));

   if (landfill_info->refuse_in_place == NULL)
      show_mem_error("refuse_periods_menu", "refuse_in_place");

   for (i = 0 ; i < total_years ; i++)
       landfill_info->refuse_in_place[i] = scratch_refuse[i];

   landfill_info->number_of_periods = total_years;
   }

/* ========================================================================== */
```



```
/*        We are done with the scratch data, free up its memory             */
/* ========================================================================== */

   if (scratch_refuse != NULL) free(scratch_refuse);
   free(year);

   sed_SetActive(data_sed, FALSE);
   sed_SetActive(scroll_sed, FALSE);

   menu_Close(data_menu);
   menu_Close(scroll_menu);

   sed_Close(data_sed);
   sed_Close(scroll_sed);
   return(ret);
}


/* ========================================================================== */
/*    This function checks the refuse period info entered by the user         */
/*                          for consistancy.                                  */
/* ========================================================================== */

int verify_data(capacity, array, count)
double *array, capacity;
int count;
{
int i;
double largest;

   largest = array[0];

   for (i = 1 ; i < count ; i++)
      if (array[i] > 0)
         {
         if (array[i] < largest)
            {
            verify_error(0);
            return(1);
            }

         largest = array[i];
         }

   if (largest > capacity)
      {
      verify_error(2);
      return(1);
      }

   if (array[count - 1] <= 0.0)
      {
      verify_error(1);
      return(1);
      }

/* ========================================================================== */
/*  Because this function is called inside the main loop which terminates     */
/*  on a return value of zero without copying the data, this function         */
/*  must return an FN10 in order for the function to continue properly.       */
/* ========================================================================== */

   return(FN10);
}
```

```
/* ================================================================== */
/*              This function rationalizes the period data.           */
/* ================================================================== */

int calculate_closure_year(landfill_info)
struct landfill_parms *landfill_info;
{
int i, j, total_years, span, lower_x, upper_x;
double lower_y, upper_y, sum = 0, last_rate, additional_years, ratio;
double remaining;

#ifdef DEBUG

   printf("Calculating closure year");
   getchar();

#endif

   total_years = landfill_info->number_of_periods;

/* ================================================================== */
/*              The rationale here bears some explanation.            */
/*                                                                    */
/* The period list represents the operating periods of the landfill   */
/* starting the next year after opening and continuing until the current */
/* year.  The first year of operation is year zero of the list.  However, */
/* it is permitted to have zero values for any of the years of operation */
/* as long as enough information is given to allow calculation of the */
/* intervening years.  Consequently, this first year may be zero and one */
/* of the later years may be non-zero.                                */
/*                                                                    */
/* To deal with the problem, we define the first point of the linear  */
/* interpolation line as being the point with index -1 and having a value */
/* of zero.  The count of zero points is started with 0.  When the first */
/* non-zero value is encountered, a for loop calculates the value of each */
/* zero point in between the non-zero value and the imaginary -1 point. */
/* Note that the index for the refuse in place point is defined as    */
/* plus 1.  When the loop finally ends, the last value calculated is one */
/* ahead of the lower_x point, ie., 0 for the first case. The zero value */
/* count is reset to 0 and lower_x and lower_y are set to  the current */
/* value.                                                             */
/* ================================================================== */

   lower_x = -1;
   lower_y = 0.0;

#ifdef DEBUG

   printf("Interpolating closure year over past history");
   getchar();

#endif

   for (i = j = 0 ; i < total_years ; i++)
      {
      if (landfill_info->refuse_in_place[i] <= 0.0) j++;
      else
         {
         span = j + 1;

         for (;j > 0 ; j--)
            landfill_info->refuse_in_place[lower_x + j] =
               lower_y + (landfill_info->refuse_in_place[i] - lower_y)
                  * j / span;

         lower_x = i;
         lower_y = landfill_info->refuse_in_place[i];
         j = 0;
         }
      }
```



```
/* ================================================================== */
/*    Calculate the average acceptance rate from the interpolated data. */
/* ================================================================== */
#ifdef DEBUG

   printf("Determining average acceptance rate");
   getchar();

#endif

   sum = landfill_info->refuse_in_place[0];
   for (i = 1 ; i < total_years ; i++)
      sum += (landfill_info->refuse_in_place[i] -
                  landfill_info->refuse_in_place[i - 1]);

   landfill_info->avg_acceptance_rate = sum / (total_years);

/* ================================================================== */
/*  Get the most recent acceptance rate.  This value will be used to  */
/*  determine the probable closing date.                              */
/* ================================================================== */

#ifdef DEBUG

   printf("Determining most recent acceptance rate");
   getchar();

#endif

   for (i = total_years - 1 ; i > 0 ; i--)
      {
      last_rate = landfill_info->refuse_in_place[i] -
         landfill_info->refuse_in_place[i - 1];

      if (last_rate > 0.0) break;
      }

   if (last_rate <= 0.0) last_rate = landfill_info->refuse_in_place[0];

   remaining = landfill_info->capacity -
      landfill_info->refuse_in_place[total_years - 1];

   if (remaining < last_rate)
      landfill_info->last_rate = remaining;
   else
      landfill_info->last_rate = last_rate;

/* ================================================================== */
/*  Determine the closure year from the last rate and the design capacity */
/* ================================================================== */

#ifdef DEBUG

   printf("Calculating years to closure (ratio)");
   getchar();

#endif

   ratio = (landfill_info->capacity -
            landfill_info->refuse_in_place[total_years - 1]) / last_rate;
```

```
#ifdef DEBUG

   printf("Rounding up");
   getchar();

#endif

   additional_years = ceil(ratio);

   if ((additional_years - ratio) > 0.9999)
      additional_years -= 1.000000;

#ifdef DEBUG

   printf("Reasoning closure");
   getchar();

#endif

   if (landfill_info->refuse_in_place[i] == landfill_info->capacity)
      {
#ifdef DEBUG

   printf("Capacity identical to last rate");
   getchar();

#endif
      landfill_info->year_closed = landfill_info->year_opened + i + 1;
      }
   else
      {
#ifdef DEBUG

   printf("Must extrapolate to closure year");
   getchar();

#endif

#ifdef DEBUG

   printf("Set closure year - is it beyond design capacity");
   getchar();

#endif

      if (additional_years > MAX_DURATION)
         {
#ifdef DEBUG

   printf("Beyond design capacity - bring it in reason");
   getchar();

#endif

            landfill_info->year_closed = landfill_info->year_opened
                                         + MAX_DURATION;

#ifdef DEBUG

   printf("Determining last rate value");
   getchar();

#endif

            landfill_info->last_rate = landfill_info->capacity
                    - landfill_info->refuse_in_place[total_years - 1];
```



```
         }
      else
         {
         landfill_info->year_closed =
            landfill_info->current_year + (int) additional_years;
         }
      }

#ifdef DEBUG

   printf("Has year closed been encountered already?");
   getchar();

#endif

   if (landfill_info->current_year >= landfill_info->year_closed)
      landfill_info->last_rate = 0.0;

#ifdef DEBUG

   printf("Calculated closure year ok - proceed to display");
   getchar();

#endif

   return(0);
}
```

```
/* ========================================================================= */
/*                                SAVEMENU.C                                 */
/* ========================================================================= */
/*    This file contains the code for displaying the save study menu.        */
/* ========================================================================= */
/*                            Created under Contract                         */
/*                                     to                                    */
/*                              EPA/AEERL/GECD/EMB                           */
/*                                                                           */
/*                                     by                                    */
/*                                                                           */
/*                              Radian Corporation                           */
/*                           Research Triangle Park, NC                      */
/* ========================================================================= */
/* ========================================================================= */
/*                            Revision History                               */
/* ========================================================================= */
/*    07/03/90   rlb       Initial Implementation                            */
/* ========================================================================= */

#include <stdio.h>
#include <cscape.h>
#include <scancode.h>
#include "helpasgn.h"
#include "colors.h"

int display_save_menu()
{
menu_type   menu, key_menu;
sed_type    sed, key_sed;
int         ret;

   menu = menu_Open();

   menu_Printf(menu, "@p[0,0]@c[%c]+@[26,-]+", menu_color);
   menu_Printf(menu, "@p[1,0]|  Do you wish to save the |");
   menu_Printf(menu, "@p[2,0]|");
   menu_Printf(menu, "@p[2,7]current study?");
   menu_Printf(menu, "@p[2,27]|");
   menu_Printf(menu, "@p[3,0]|@[26,-]|");
   menu_Printf(menu, "@p[4,0]|");
   menu_Printf(menu, "@p[4,27]|");
   menu_Printf(menu, "@p[5,0]|");
   menu_Printf(menu, "@p[5,27]|");
   menu_Printf(menu, "@p[6,0]|");
   menu_Printf(menu, "@p[6,27]|");
   menu_Printf(menu, "@p[7,0]|");
   menu_Printf(menu, "@p[7,27]|");
   menu_Printf(menu, "@p[8,0]+@[26,-]+");
   menu_Printf(menu, "@p[5,1]@f[@[11, ]Yes@[12, ]]",
     NULL, &menu_funcs);
   menu_Printf(menu, "@p[6,1]@f[@[11, ]No@[13, ]]",
     NULL, &menu_funcs);

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, menu_color, menu_color, field_color);
   sed_SetPosition(sed, 8, 25);

   sed_Push(sed);
   sed_Repaint(sed);
   show_key_help(&key_menu, &key_sed, abortable_menu_help_string);

   do
```



```
      |
      ret = sed_Go(sed);

      if (ret == FN1)
         |
         compile_help(DISPLAY_SAVE_MENU);
         sed_SetBaton(sed, -1);
         |
      }
   while (ret > sed_GetFieldCount(sed));

   destroy_key_help(key_menu, key_sed);
   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);
   return(ret);
}
```

```
/* ========================================================================== */
/*                                  SAVESTUD.C                                */
/* ========================================================================== */
/*  This file contains the code for saving the study information to a         */
/*  file set.                                                                 */
/* ========================================================================== */
/*                             Created under Contract                         */
/*                                       to                                   */
/*                               EPA/AEERL/GECD/EMB                           */
/*                                                                            */
/*                                       by                                   */
/*                                                                            */
/*                               Radian Corporation                           */
/*                            Research Triangle Park, NC                      */
/* ========================================================================== */
/* ========================================================================== */
/*                             Revision History                               */
/* ========================================================================== */
/*    07/03/90    rlb       Initial Implementation                            */
/*    01/17/94    rlb       Added support for saving codisposal and           */
/*                             no codisposal factors to save set.             */
/* ========================================================================== */

#include <dos.h>
#include <stdio.h>
#include <cscape.h>
#include <string.h>
#include <scancode.h>
#include "landfill.h"
#include "landuser.h"
#include "helpasgn.h"
#include "colors.h"


int save_study_info(name, landfill, model, calc_possible)
char *name;
struct landfill_parms landfill;
struct model_parms model;
boolean calc_possible;
{
int i, j;
FILE *parameters, *airtoxics;
char string[256];
extern struct system_parms config;
unsigned attributes;

   strcpy(string, config.working_dir);
   strcat(string, "\\");
   strcat(string, name);
   strcat(string, ".prm");

   if (! _dos_getfileattr(string, &attributes))
      if ((replace_study() == 1) ? 0 : 1) return(-1);

   parameters = fopen(string, "w");

   if (parameters == NULL)
      {
      show_file_error();
      return(-1);
      }

   fprintf(parameters, "%f %f %f %f %f %d\n", model.l0, model.k,
                                              model.nmoc,
                                              model.ch4, model.co2,
                                              (int) model.include_air_toxics);
```



```
   fprintf(parameters, "%d %d %d\n", landfill.year_opened,
                                     landfill.current_year,
                                     landfill.year_closed);

   fprintf(parameters, "%f %f %f\n", landfill.capacity,
                                     landfill.avg_acceptance_rate,
                                     landfill.last_rate);

   fprintf(parameters, "%d\n", (int) calc_possible);

   fprintf(parameters, "%d\n", landfill.number_of_periods);

   for (i = 0 ; i < landfill.number_of_periods ; i++)
      {
      if (fprintf(parameters, "%f\n", landfill.refuse_in_place[i]) < 0)
         {
         show_file_error();
         return(-1);
         }
      }

   if (ferror(parameters))
      {
      show_file_error();
      return(-1);
      }

   fclose(parameters);

   strcpy(string, config.working_dir);
   strcat(string, "\\");
   strcat(string, name);
   strcat(string, ".atx");

   airtoxics = fopen(string, "w");

   if (airtoxics == NULL)
      {
      show_file_error();
      return(-1);
      }

   for (i = 0 ; i < model.number_air_toxics ; i++)
      {
      for (j = 0 ; ((j < 30) &&
                    (model.air_toxics[i].chemical_name[j] != '\0')) ; j++)
         string[j] = model.air_toxics[i].chemical_name[j];

      for (; j < 30 ; j++)
         string[j] = ' ';

      string[j] = '\0';

      sprintf(string + j, "%f %f %f", model.air_toxics[i].molecular_wt,
                                      model.air_toxics[i].concentration[NOCODISPOSA
L],
                                      model.air_toxics[i].concentration[CODISPOSAL]
);

      if (fprintf(airtoxics, "%s\n", string) < 0)
         {
         show_file_error();
         return(-1);
         }
      }
```

```
   if (ferror(airtoxics))
      {
      show_file_error();
      return(-1);
      }

   fclose(airtoxics);

   return(0);
}


int replace_study()
{
menu_type   menu, key_menu;
sed_type   sed, key_sed;
int       ret;

   menu = menu_Open();

   buildbox(menu, menu_color, 9, 20);
   menu_Printf(menu, "@p[1,4]Study Already Exists");
   menu_Printf(menu, "@p[2,0]¦          Replace?          ¦");
   menu_Printf(menu, "@p[3,0]¦@[26,-]¦");
   menu_Printf(menu, "@p[5,1]@f[@[11, ]Yes@(12, ]]",
     NULL, &menu_funcs);
   menu_Printf(menu, "@p[6,1]@f[@[11, ]No@(13, ]]",
     NULL, &menu_funcs);

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, menu_color, menu_color, field_color);
   sed_SetPosition(sed, 8, 25);

   sed_Push(sed);
   sed_Repaint(sed);
   show_key_help(&key_menu, &key_sed, abortable_menu_help_string);

   do
      {
      ret = sed_Go(sed);

      if (ret == FN1)
         compile_help(DISPLAY_SAVE_MENU);
      }
   while (ret > sed_GetFieldCount(sed));

   destroy_key_help(key_menu, key_sed);
   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);
   return(ret);
}
```

```
/* ========================================================================== */
/*                                 SCR_VIEW.C                                 */
/* ========================================================================== */
/*  This file contains the code for displaying the results tabularly          */
/* ========================================================================== */
/*                            Created under Contract                          */
/*                                    to                                      */
/*                              EPA/AEERL/GECD/EMB                            */
/*                                                                            */
/*                                    by                                      */
/*                                                                            */
/*                              Radian Corporation                            */
/*                           Research Triangle Park, NC                       */
/* ========================================================================== */
/* ========================================================================== */
/*                             Revision History                               */
/* ========================================================================== */
/*    07/03/90   rlb       Initial Implementation                             */
/*    01/21/94   rlb       Changed routine to properly use the air toxics     */
/*                           flag for codisposal option.                      */
/* ========================================================================== */

#include <stdio.h>
#include <cscape.h>
#include <scancode.h>
#include <string.h>
#include <stdlib.h>
#include <errno.h>
#include "landfill.h"
#include "defaults.h"
#include "constant.h"
#include "helpasgn.h"
#include "colors.h"

/* ========================================================================== */
/*      IBM specific values for character codes for various symbols.          */
/* ========================================================================== */

#define UP_ARROW      24
#define DOWN_ARROW    25
#define SPACE         32


int show_results(model_info, landfill_info, results, type)
struct landfill_parms *landfill_info;
struct model_parms *model_info;
struct model_results *results;
int type;
{
int old_cursor;
menu_type   menu, view_menu, key_menu;
sed_type    sed, view_sed, key_sed;
int       ret, line = 0, i, j;
char up_indicator = SPACE, down_indicator = SPACE, string[256];
int curr_offset = 0 , old_offset = 0;
double sum = 0;

extern double mass_to_volume(double, double, double, double);
extern double volume_to_mass(double, double, double, double);

   old_cursor = vid_GetCursorType();
   vid_SetCursorType(CURSOR_NONE);

/* ========================================================================== */
/*                            Build the view box                              */
```



```
/* ========================================================================== */

   errno = 0;

   menu = menu_Open();

   if (menu == NULL)
      {
      show_mem_error4();
      return(0);
      }

   build_scrollbox(menu, info_color, 24, 80);

   menu_Printf(menu, "@p[1,32]View Model Results");
   menu_Printf(menu, "@p[10,20]Preparing Tabular Report, Please Wait ...");

   menu_Printf(menu, "@p[3,1]@f[#]",
     &down_indicator, &char_funcs);
   menu_Printf(menu, "@p[22,1]@f[#]",
     &up_indicator, &char_funcs);

   if (errno == ENOMEM)
      {
      if (menu != NULL) menu_Close(menu);
      show_mem_error4();

      return(0);
      }

   menu_Flush(menu);

   sed = sed_Open(menu);

   if (sed == NULL)
      {
      if (menu != NULL) menu_Close(menu);
      show_mem_error4();

      return(0);
      }

   sed_SetColors(sed, info_color, info_color, info_color);
   sed_SetPosition(sed, 0, 0);

   sed_Push(sed);

   sed_Repaint(sed);
   show_key_help(&key_menu, &key_sed, tabular_report_key_help);

/* ========================================================================== */
/*                    Build the information to be viewed.                     */
/* ========================================================================== */

   view_menu = menu_Open();

   if (view_menu == NULL)
      {
      sed_Pop(sed);
      if (menu != NULL) menu_Close(menu);
      if (sed != NULL) sed_Close(sed);

      show_mem_error4();
      return(0);
      }
```

```
   menu_Printf(view_menu, "@[0,0]@c[%c]@[76, ]", info_color);

   menu_Printf(view_menu, "@p[%d,29]@c[%c]Model Parameters", ++line,
               info_color);
   menu_Printf(view_menu, "@p[%d,5]@[66,=]", ++line);

   ++line;

   sprintf(string, "@p[%d,5]Lo : %f Cft / Mg", ++line, model_info->l0);

   if (((config.default_mode == AP_MODE)
         && (model_info->l0 != DEFAULT_AP42_L0)) ||
       ((config.default_mode == REGULATORY_MODE)
         && (model_info->l0 != DEFAULT_REGULATORY_L0)))
      strcat(string, "  **** Note - Default value not used ****");

   menu_Printf(view_menu, string);

   sprintf(string, "@p[%d,5]k : %f 1/yr", ++line, model_info->k);

   if (((config.default_mode == AP_MODE)
         && (model_info->k != DEFAULT_AP42_K)) ||
       ((config.default_mode == REGULATORY_MODE)
         && (model_info->k != DEFAULT_REGULATORY_K)))
      strcat(string, "  **** Note - Default value not used ****");

   menu_Printf(view_menu, string);

   ++line;

   sprintf(string, "@p[%d,5]NMOC : %f ppmv", ++line, model_info->nmoc);

   if (config.default_mode == AP_MODE)
      {
      if (((model_info->include_air_toxics == CODISPOSAL)
            && (model_info->nmoc != DEFAULT_AP42_NMOC_CODIS)) ||
          ((model_info->include_air_toxics == NOCODISPOSAL)
            && (model_info->nmoc != DEFAULT_AP42_NMOC_NOCO)))
         strcat(string, "  **** Note - Default value not used ****");
      }
   else
      {
      if (model_info->nmoc != DEFAULT_REGULATORY_NMOC)
         strcat(string, "  **** Note - Default value not used ****");
      }

   menu_Printf(view_menu, string);

   sprintf(string, "@p[%d,5]Methane : %f %% volume", ++line,
                   model_info->ch4);
   if (model_info->ch4 != DEFAULT_CH4)
      strcat(string, "  **** Note - Default value not used ****");
   menu_Printf(view_menu, string);

   menu_Printf(view_menu, "@p[%d,5]Carbon Dioxide : %f %% volume", ++line,
                  model_info->co2);

   line++;

   if ((model_info->include_air_toxics == NO_AT) && (type  > 3))
      {
      menu_Printf(view_menu, "@p[%d,5]Individual Air Toxics Analysis Not Selecte
d",
                     ++line);
      ++line;
      }
```



```
   if ((model_info->include_air_toxics != NO_AT) && (type > 3))
      {
      menu_Printf(view_menu, "@p[%d,5]Desired Air Toxic Parameters : %s",
                     ++line, model_info->air_toxics[type - 4].chemical_name);
      sprintf(string, "@p[%d,5]Molecular Wt = %8.2f     Concentration = %13.6f",
                  ++line, model_info->air_toxics[type - 4].molecular_wt,
                     model_info->air_toxics[type - 4].concentration[model_inf
o->include_air_toxics]);

      menu_Printf(view_menu, string);
      }

   ++line;

   menu_Printf(view_menu, "@p[%d,27]Landfill Parameters", ++line);
   menu_Printf(view_menu, "@p[%d,5]@[66,=]", ++line);
   ++line;

   menu_Printf(view_menu, "@p[%d,5]Year Opened : %d    Current Year : %d    Year
 Closed : %d",
                  ++line, landfill_info->year_opened, landfill_info->current_yea
r,
                  landfill_info->year_closed);

   menu_Printf(view_menu, "@p[%d,5]Capacity : %f  Mg", ++line,
                  landfill_info->capacity);
   menu_Printf(view_menu, "@p[%d,5]Average Acceptance Rate : %f  Mg/year",
                  ++line, landfill_info->avg_acceptance_rate);
   menu_Printf(view_menu, "@p[%d,5]Average Acceptance Rate Required from", ++lin
e);
   menu_Printf(view_menu, "@p[%d,5]   Current Year to Closure Year :  %f Mg/year
",
                  ++line, landfill_info->last_rate);

   ++line;

   if (((model_info->include_air_toxics != NO_AT) && (type  > 3)) || (type < 3))
      {
      menu_Printf(view_menu, "@p[%d,29]Model Results", ++line);
      menu_Printf(view_menu, "@p[%d,5]@[66,=]", ++line);
      ++line;
      }

   if (errno == ENOMEM)
      {
      sed_Pop(sed);
      if (menu != NULL) menu_Close(menu);
      if (sed != NULL) sed_Close(sed);

      show_mem_error4();
      return(0);
      }

   switch (type)
      {
      case 1:
         menu_Printf(view_menu, "@p[%d,5]
    Methane Emission Rate", ++line);
         menu_Printf(view_menu, "@p[%d,5]Year      Refuse In Place (Mg)
(Mg/yr)       (Cubic Ft/yr)", ++line);
         menu_Printf(view_menu, "@p[%d,5]@[66,=]", ++line);

         for (sum = i = 0 ; i < results->year_count ; i++)
            {
            sum += results->refuse_data[i];
```

```
            sprintf(string, "@p[%d,5]%4d        %13.3E            %12.3E       %12
.3E", ++line,
                        landfill_info->year_opened + 1 + i, sum,
                        (volume_to_mass(results->annual_ch4[i] * FT3_TO_M3,
                                        16.04, DEFAULT_TEMP,
                                        DEFAULT_PRESSURE) / 1000.0),
                        results->annual_ch4[i]);

            menu_Printf(view_menu, string);

            if (errno == ENOMEM)
               {
               sed_Pop(sed);
               if (menu != NULL) menu_Close(menu);
               if (sed != NULL) sed_Close(sed);

               show_mem_error4();
               return(0);
               }
            ]

         ++line;
         break;

      case 2:
         menu_Printf(view_menu, "@p[%d,5]                                        C
arbon Dioxide Emission Rate", ++line);
         menu_Printf(view_menu, "@p[%d,5]Year      Refuse In Place (Mg)
(Mg/yr)       (Cubic Ft/yr)", ++line);
         menu_Printf(view_menu, "@p[%d,5]@[66,=]", ++line);

         for (sum = i = 0 ; i < results->year_count ; i++)
            {
            sum += results->refuse_data[i];
            sprintf(string, "@p[%d,5]%4d        %13.3E            %12.3E       %1
2.3E", ++line,
                        landfill_info->year_opened + 1 + i, sum,
                        (volume_to_mass(results->annual_co2[i] * FT3_TO_M3,
                                        44.01, DEFAULT_TEMP,
                                        DEFAULT_PRESSURE) / 1000.0),
                        results->annual_co2[i]);

            menu_Printf(view_menu, string);

            if (errno == ENOMEM)
               {
               sed_Pop(sed);
               if (menu != NULL) menu_Close(menu);
               if (sed != NULL) sed_Close(sed);

               show_mem_error4();
               return(0);
               }
            }

         ++line;
         break;

      case 3:
         menu_Printf(view_menu, "@p[%d,5]                                        NM
OC Emission Rate (as Hexane)", ++line);
         menu_Printf(view_menu, "@p[%d,5]Year      Refuse In Place (Mg)
(Mg/yr)       (Cubic Ft/yr)", ++line);
         menu_Printf(view_menu, "@p[%d,5]@[66,=]", ++line);

         for (sum = i = 0 ; i < results->year_count ; i++)
```



```
            {
            sum += results->refuse_data[i];
            sprintf(string, "@p[%d,5]%4d        %13.3E            %12.3E       %1
2.3E", ++line,
                        landfill_info->year_opened + 1 + i, sum,
                        results->annual_nmoc[i],
                        (mass_to_volume(results->annual_nmoc[i] * 1000.0,
                         DEFAULT_NMOC_MOLE_WT, DEFAULT_TEMP,
                         DEFAULT_PRESSURE) / FT3_TO_M3));

            menu_Printf(view_menu, string);

            if (errno == ENOMEM)
               {
               sed_Pop(sed);
               if (menu != NULL) menu_Close(menu);
               if (sed != NULL) sed_Close(sed);

               show_mem_error4();
               return(0);
               }
            ]

         ++line;
         break;

      default:
         ++line;

         if (model_info->include_air_toxics != NO_AT)
            {
            j = type - 4;

            menu_Printf(view_menu, "@p[%d,15]Selected Air Toxic: %s",
                        ++line, model_info->air_toxics[j].chemical_name);
            menu_Printf(view_menu, "@p[%d,5]@[66,=]", ++line);

            if (model_info->air_toxics[j].concentration[model_info->include_air_
toxics] > 0.0)
               {
               ++line;

               menu_Printf(view_menu, "@p[%d,5]Year      Refuse In Place (Mg)
      (Mg/yr)       (Cubic Ft/yr)", ++line);
               menu_Printf(view_menu, "@p[%d,5]@[66,=]", ++line);

               for (sum = i = 0 ; i < results->year_count ; i++)
                  {
                  sum += results->refuse_data[i];

                  sprintf(string, "@p[%d,5]%4d        %13.3E            %12.3E
   %12.3E", ++line,
                              landfill_info->year_opened + 1 + i, sum,
                              (volume_to_mass(results->annual_air_toxics[j][i] *
 FT3_TO_M3,
                                              model_info->air_toxics[j].molecula
r_wt,
                                              DEFAULT_TEMP,
                                              DEFAULT_PRESSURE) / 1000.0),
                              results->annual_air_toxics[j][i]);

                  menu_Printf(view_menu, string);

                  if (errno == ENOMEM)
                     {
                     sed_Pop(sed);
```

```
                    if (menu != NULL) menu_Close(menu);
                    if (sed != NULL) sed_Close(sed);

                    show_mem_error4();
                    return(0);
                    }
                }
            }
         else
            {
            ++line;
            menu_Printf(view_menu, "@p[%d,23]Selection Results were all zero"
,
                        ++line);
            }
         }
      }

   menu_Flush(view_menu);

   if (errno == ENOMEM)
      {
      sed_Pop(sed);

      if (view_menu != NULL) menu_Close(view_menu);
      if (menu != NULL) menu_Close(menu);
      if (sed != NULL) sed_Close(sed);

      show_mem_error4();
      return(0);
      }

   view_sed = sed_Open(view_menu);

   if (view_sed == NULL)
      {
      if (view_menu != NULL) menu_Close(view_menu);
      if (menu != NULL) menu_Close(menu);
      if (sed != NULL) sed_Close(sed);

      show_mem_error4();

      return(0);
      }

   sed_SetColors(view_sed, info_color, info_color, info_color);
   sed_SetPosition(view_sed, 3, 3);

   sed_SetWidth(view_sed, 76);
   sed_SetHeight(view_sed, 20);

   if (line > 20) down_indicator = DOWN_ARROW;

   sed_RepaintFields(sed);
   sed_Repaint(view_sed);

   do
      {
      ret = kb_Read();

      switch (ret)
         {
         case FN1:
            compile_help(SHOW_RESULTS);
            break;
```



```
         case PGUP:
            sed_ScrollUp(view_sed, 18);
            break;

         case PGDN:
            sed_ScrollDown(view_sed, 18);
            break;

         case UP:
            sed_ScrollUp(view_sed, 1);
            break;

         case DOWN:
            sed_ScrollDown(view_sed, 1);
            break;
         }

      curr_offset = sed_GetYoffset(view_sed);

      if (curr_offset) up_indicator = UP_ARROW;
      else up_indicator = SPACE;

      if ((line - curr_offset) > 20) down_indicator = DOWN_ARROW;
      else down_indicator = SPACE;

      if (curr_offset != old_offset)
         {
         sed_RepaintFields(sed);
         old_offset = curr_offset;
         }
      }
   while ((ret != ESC) && (ret != FN10));

   destroy_key_help(key_menu, key_sed);
   sed_Pop(sed);

   menu_Close(view_menu);
   menu_Close(menu);

   sed_Close(view_sed);
   sed_Close(sed);

   vid_SetCursorType(old_cursor);

   return(ret);
}
```

```
/* ========================================================================== */
/*                                  SELECTOR.C                                */
/* ========================================================================== */
/*  This file contains the code for selecting the chemical to report.         */
/* ========================================================================== */
/*                            Created under Contract                          */
/*                                      to                                    */
/*                              EPA/AEERL/GECD/EMB                            */
/*                                                                            */
/*                                      by                                    */
/*                                                                            */
/*                              Radian Corporation                            */
/*                           Research Triangle Park, NC                       */
/* ========================================================================== */
/* ========================================================================== */
/*                              Revision History                              */
/* ========================================================================== */
/*    07/03/90   rlb       Initial Implementation                             */
/* ========================================================================== */

#include <stdio.h>
#include <cscape.h>
#include <scancode.h>
#include <string.h>
#include "landfill.h"
#include "helpasgn.h"
#include "colors.h"

#define UP_ARROW      24
#define DOWN_ARROW    25
#define SPACE         32

int report_selector(air_toxics, count, all_selection)
struct air_toxic_structure *air_toxics;
int count;
char *all_selection;
{
menu_type   menu, list_menu, key_menu;
sed_type    sed, list_sed, key_sed;
int         i, ret, right_pad, curr_offset = 0, old_offset;
int down_indicator = SPACE, up_indicator = SPACE;
char format[80];

   if ((count + 3) > 7) down_indicator = DOWN_ARROW;

   menu = menu_Open();

   menu_Printf(menu, "@p[0,0]@c[%c]+@[39,-]+", menu_color);
   menu_Printf(menu, "@p[1,0]|");
   menu_Printf(menu, "@p[1,7]Choose a Chemical to Report");
   menu_Printf(menu, "@p[1,40]|");
   menu_Printf(menu, "@p[2,0]|--@[37,-]|");
   menu_Printf(menu, "@p[3,0]| |");
   menu_Printf(menu, "@p[3,40]|");
   menu_Printf(menu, "@p[4,0]| |");
   menu_Printf(menu, "@p[4,40]|");
   menu_Printf(menu, "@p[5,0]| |");
   menu_Printf(menu, "@p[5,40]|");
   menu_Printf(menu, "@p[6,0]| |");
   menu_Printf(menu, "@p[6,40]|");
   menu_Printf(menu, "@p[7,0]| |");
   menu_Printf(menu, "@p[7,40]|");
   menu_Printf(menu, "@p[8,0]| |");
   menu_Printf(menu, "@p[8,40]|");
   menu_Printf(menu, "@p[9,0]| |");
```



```
   menu_Printf(menu, "@p[9,40]|");
   menu_Printf(menu, "@p[10,0]| |");
   menu_Printf(menu, "@p[10,40]|");
   menu_Printf(menu, "@p[11,0]+--@[37,-]+");

   menu_Printf(menu, "@p[3,1]@fp[#]", &down_indicator, &char_funcs);
   menu_Printf(menu, "@p[10,1]@fp[#]", &up_indicator, &char_funcs);

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, menu_color, menu_color, menu_color);
   sed_SetPosition(sed, 6, 19);

   list_menu = menu_Open();

   menu_Printf(list_menu, "@p[0,0]@c[%c]@f[ Methane@[29, ]]", menu_color,
               NULL, &menu_funcs);
   menu_Printf(list_menu, "@p[1,0]@f[ Carbon Dioxide@[23, ]]", NULL,
               &menu_funcs);
   menu_Printf(list_menu, "@p[2,0]@f[ NMOC@[33, ]]", NULL,
               &menu_funcs);

   for (i = 0 ; i < 8 ; i++)
      menu_Printf(list_menu, "@p[%d,0]@[37, ]", i);

   for (i = 0 ; i < count ; i++)
      {
      right_pad = 36 - strlen(air_toxics[i].chemical_name);

      sprintf(format, "@p[%d,0]@c[%c]@f[ %s@[%d, ]]", i + 3, menu_color,
                      air_toxics[i].chemical_name,
                      (right_pad >= 0 ? right_pad : 0));

      menu_Printf(list_menu, format, NULL, &menu_funcs);
      }

   if (all_selection != NULL)
      {
      right_pad = 36 - strlen(all_selection);

      sprintf(format, "@p[%d,0]@c[%c]@f[ %s@[%d, ]]", i + 3, menu_color,
                      all_selection,
                      (right_pad >= 0 ? right_pad : 0));

      menu_Printf(list_menu, format, NULL, &menu_funcs);
      }

   menu_Flush(list_menu);

   list_sed = sed_Open(list_menu);
   sed_SetColors(list_sed, menu_color, menu_color, field_color);
   sed_SetPosition(list_sed, 9, 22);

   sed_SetWidth(list_sed, 37);
   sed_SetHeight(list_sed, 8);

   sed_Push(sed);
   sed_Repaint(sed);
   show_key_help(&key_menu, &key_sed, generic_menu_help_string);

   sed_Repaint(list_sed);

   sed_SetActive(list_sed, TRUE);

   sed_GotoFirstField(list_sed);
```

```
   sed_DoFieldSenter(list_sed, sed_GetFieldNo(list_sed));
   sed_DoFieldFenter(list_sed, sed_GetFieldNo(list_sed));

   do
      {
      sed_DoFieldFkey(list_sed, sed_GetFieldNo(list_sed));

      ret = sed_GetBaton(list_sed);

      switch (ret)
         {
         case FN1:
            compile_help(REPORT_SELECTOR);
            sed_SetBaton(list_sed, -1);
            break;
         }

      curr_offset = sed_GetYoffset(list_sed);

      if (curr_offset) up_indicator = UP_ARROW;
      else up_indicator = SPACE;

      if ((curr_offset + 8) < (count + 3)) down_indicator = DOWN_ARROW;
      else down_indicator = SPACE;

      if (curr_offset != old_offset)
         {
         sed_RepaintFields(sed);
         old_offset = curr_offset;
         }
      }
   while ((ret > sed_GetFieldCount(list_sed)) || (ret == -1));

   destroy_key_help(key_menu, key_sed);
   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);

   menu_Close(list_menu);
   sed_Close(list_sed);

   return(ret);
}
```

```
/* ========================================================================== */
/*                                  STRMVIEW.C                                */
/* ========================================================================== */
/*    This file contains the code for printing the report on a stream.        */
/* ========================================================================== */
/*                             Created under Contract                         */
/*                                       to                                   */
/*                               EPA/AEERL/GECD/EMB                           */
/*                                                                            */
/*                                       by                                   */
/*                                                                            */
/*                                 Radian Corporation                         */
/*                            Research Triangle Park, NC                      */
/* ========================================================================== */
/* ========================================================================== */
/*                                Revision History                            */
/* ========================================================================== */
/*    07/03/90   rlb       Initial Implementation                             */
/*    01/21/94   rlb       Change to allow use of include_air_toxics flag     */
/*                            as the selection of codisposal option.          */
/*    02/08/94   rlb       Changed to permit default selection.               */
/* ========================================================================== */

#include <stdio.h>
#include <cscape.h>
#include <string.h>
#include "landfill.h"
#include "defaults.h"
#include "constant.h"
#include "colors.h"


int print_results(dir, filename, model_info, landfill_info, results, type)
char *dir, *filename;
struct model_parms *model_info;
struct landfill_parms *landfill_info;
struct model_results *results;
int type;
{
FILE *stream, *err;
char string[PAGE_WIDTH + 5];
int i, j;

void sub_reports();

   if (dir != NULL)
      {
      strcpy(string, dir);
      strcat(string, "\\");
      strcat(string, filename);

      stream = fopen(string, "w");
      }
   else
      stream = fopen(filename, "w");

   if (stream == NULL)
      {
      print_error(filename);
      return(-1);
      }
   else
      print_go(filename);

   sprintf(string, "Model Parameters");
   fprintf(stream, "%s\n", strcenter(string, PAGE_WIDTH));
```



```
   if (ferror(stream))
      {
      print_error(filename);
      return(1);
      }

   blank_string(string);
   for (i = 5 ; i < PAGE_WIDTH - 5 ; i++)
      string[i] = '=';
   string[i] = '\0';
   if (fprintf(stream, "%s\n", string) < 0) print_error(filename);

   if (fprintf(stream, "      Lo : %f ft^3 / Mg", model_info->l0) < 0)
      print_error(filename);

   if (config.default_mode == AP_MODE)
      {
      if (model_info->l0 != DEFAULT_AP42_L0)
         if (fprintf(stream, " ***** Note : Default value not used ****") < 0)
            print_error(filename);
      }
   else
      {
      if (model_info->l0 != DEFAULT_REGULATORY_L0)
         if (fprintf(stream, " ***** Note : Default value not used ****") < 0)
            print_error(filename);
      }

   fprintf(stream, "\n      k : %f 1/yr", model_info->k);

   if (config.default_mode == AP_MODE)
      {
      if (model_info->k != DEFAULT_AP42_K)
         fprintf(stream, " ***** Note : Default value not used ****");
      }
   else
      {
      if (model_info->k != DEFAULT_REGULATORY_K)
         fprintf(stream, " ***** Note : Default value not used ****");
      }

   fprintf(stream, "\n\n      NMOC : %f ppmv", model_info->nmoc);

   if (config.default_mode == AP_MODE)
      {
      if (model_info->include_air_toxics == CODISPOSAL)
         {
         if (model_info->nmoc != DEFAULT_AP42_NMOC_CODIS)
            fprintf(stream, " ***** Note : Default value not used ****");
         }
      else
         {
         if (model_info->nmoc != DEFAULT_AP42_NMOC_NOCO)
            fprintf(stream, " ***** Note : Default value not used ****");
         }
      }
   else
      {
      if (model_info->nmoc != DEFAULT_REGULATORY_NMOC)
         fprintf(stream, " ***** Note : Default value not used ****");
      }

   fprintf(stream, "\n      Methane : %f %% volume", model_info->ch4);
   if (model_info->ch4 != DEFAULT_CH4)
      fprintf(stream, " ***** Note : Default value not used ****");
```

```
   fprintf(stream, "\n     Carbon Dioxide : %f %% volume\n\n", model_info->co2);

   if ((model_info->include_air_toxics != NO_AT) && (type > 3))
      {
      fprintf(stream, "     Desired Air Toxic Parameters : %s\n",
         model_info->air_toxics[type - 4].chemical_name);

      fprintf(stream, "     Molecular Wt = %8.2f     Concentration = %13.6f\n\n"
,
                  model_info->air_toxics[type - 4].molecular_wt,
                  model_info->air_toxics[type-4].concentration[model_info->inclu
de_air_toxics]);
      }

   if (type == 0)
      {
      sprintf(string, "Air Toxics Parameters");
      fprintf(stream, "%s\n", strcenter(string, PAGE_WIDTH));

      fprintf(stream, "     Chemical Name                              Molecular Wt
    Concentration\n");

      blank_string(string);
      for (i = 5 ; i < PAGE_WIDTH - 5 ; i++)
         string[i] = '=';
      string[i] = '\0';
      fprintf(stream, "%s\n", string);

      for (i = 0 ; i < model_info->number_air_toxics ; i++)
         {
         blank_string(string);
         sprintf(string, "     %s", model_info->air_toxics[i].chemical_name);

         for (j = strlen(string) ; j < 35 ; j++)
            string[j] = ' ';
         string[j] = '\0';

         sprintf(string + 35, "       %13.3f       %13.3f",
                     model_info->air_toxics[i].molecular_wt,
                     model_info->air_toxics[i].concentration[model_info->include
_air_toxics]);

         fprintf(stream, "%s\n", string);
         }
      }

   sprintf(string, "Landfill Parameters");
   fprintf(stream, "\n%s\n", strcenter(string, PAGE_WIDTH));

   blank_string(string);
   for (i = 5 ; i < PAGE_WIDTH - 5 ; i++)
      string[i] = '=';
   string[i] = '\0';
   fprintf(stream, "%s\n", string);

   fprintf(stream, "     Year Opened : %d    Current Year : %d   Year Closed: %d
\n",
               landfill_info->year_opened, landfill_info->current_year,
               landfill_info->year_closed);

   fprintf(stream, "     Capacity : %f Mg\n", landfill_info->capacity);
   fprintf(stream, "     Average Acceptance Rate : %f  Mg/year\n",
               landfill_info->avg_acceptance_rate);
   fprintf(stream, "     Average Acceptance Rate Required from\n");
   fprintf(stream, "        Current Year to Closure Year : %f Mg/year\n\n",
```



```
               landfill_info->last_rate);

   if (((model_info->include_air_toxics != NO_AT)
        && (type > 3)) || (type < 3))
      {
      sprintf(string, "Model Parameters");
      fprintf(stream, "%s\n", strcenter(string, PAGE_WIDTH));

      blank_string(string);
      for (i = 5 ; i < PAGE_WIDTH - 5 ; i++)
         string[i] = '=';
      string[i] = '\0';
      fprintf(stream, "%s\n", string);
      }

   if (type == 0)
      {
      for (i = 1 ; i < (model_info->number_air_toxics + 4) ; i++)
         {
         sub_reports(stream, i, string, model_info, landfill_info, results);
         fprintf(stream, "\n");
         }
      }
   else
      sub_reports(stream, type, string, model_info, landfill_info, results);

   if (ferror(stream)) print_error(filename);

   fclose(stream);

   vid_Clear('\x07');
}


void sub_reports(stream, type, string, model_info, landfill_info, results)
FILE *stream;
int type;
char *string;
struct model_parms *model_info;
struct landfill_parms *landfill_info;
struct model_results *results;
{
int i, j;
double sum;

extern double mass_to_volume(double, double, double, double);
extern double volume_to_mass(double, double, double, double);

   switch (type)
      {
      case 1:
         fprintf(stream, "                                                                Me
thane Emission Rate  \n");
         fprintf(stream, "     Year     Refuse In Place (Mg)                (Mg/
yr)     (Cubic Ft/yr)\n");

         blank_string(string);
         for (i = 5 ; i < PAGE_WIDTH - 5 ; i++)
            string[i] = '=';
         string[i] = '\0';
         fprintf(stream, "%s\n", string);

         for (sum = i = 0 ; i < results->year_count ; i++)
            {
            sum += results->refuse_data[i];
            sprintf(string, "     %4d            %13.3E           %12.3E      %1
```

```
2.3E",
                        landfill_info->year_opened + 1 + i, sum,
                        (volume_to_mass(results->annual_ch4[i] * FT3_TO_M3,
                                        16.04, DEFAULT_TEMP,
                                        DEFAULT_PRESSURE) / 1000.0),
                        results->annual_ch4[i]);

                fprintf(stream, "%s\n", string);
                }

            break;

        case 2:
            fprintf(stream, "                                                    Carbon
Dioxide Emission Rate\n");
            fprintf(stream, "     Year     Refuse In Place (Mg)                (Mg/y
r)     (Cubic Ft/yr) \n");

            blank_string(string);
            for (i = 5 ; i < PAGE_WIDTH - 5 ; i++)
               string[i] = '=';
            string[i] = '\0';
            fprintf(stream, "%s\n", string);

            for (sum = i = 0 ; i < results->year_count ; i++)
               {
               sum += results->refuse_data[i];
               sprintf(string, "     %4d            %13.3E          %12.3E      %12
.3E",
                        landfill_info->year_opened + 1 + i, sum,
                        (volume_to_mass(results->annual_co2[i] * FT3_TO_M3,
                                        44.01, DEFAULT_TEMP,
                                        DEFAULT_PRESSURE) / 1000.0),
                        results->annual_co2[i]);

               fprintf(stream, "%s\n", string);
               }

            break;

        case 3:
            fprintf(stream, "                                                    NMOC Em
ission Rate (as Hexane)\n");
            fprintf(stream, "     Year     Refuse In Place (Mg)                (Mg/y
r)     (Cubic Ft/yr) \n");

            blank_string(string);
            for (i = 5 ; i < PAGE_WIDTH - 5 ; i++)
               string[i] = '=';
            string[i] = '\0';
            fprintf(stream, "%s\n", string);


            for (sum = i = 0 ; i < results->year_count ; i++)
               {
               sum += results->refuse_data[i];
               sprintf(string, "     %4d            %13.3E          %12.3E      %12
.3E",
                        landfill_info->year_opened + 1 + i, sum,
                        results->annual_nmoc[i],
                        (mass_to_volume(results->annual_nmoc[i] * 1000.0,
                        DEFAULT_NMOC_MOLE_WT, DEFAULT_TEMP,
                        DEFAULT_PRESSURE) / FT3_TO_M3));

               fprintf(stream, "%s\n", string);
               }
```



```
            break;

        default:
            if (model_info->include_air_toxics != NO_AT)
               {
               j = type - 4;

               sprintf(string, "Selected Air Toxic: %s",
                           model_info->air_toxics[j].chemical_name);
               fprintf(stream, "          %s\n",
                           strcenter(string, PAGE_WIDTH - 29));

               blank_string(string);
               for (i = 7 ; i < PAGE_WIDTH - 16 ; i++)
                  string[i] = '=';
               string[i] = '\0';
               fprintf(stream, "%s\n", string);

               if (model_info->air_toxics[j].concentration[model_info->include_air_
toxics] > 0.0)
                  [
                  fprintf(stream, "     Year     Refuse In Place (Mg)
 (Mg/yr)     (Cubic Ft/yr) \n");

                  blank_string(string);
                  for (i = 5 ; i < PAGE_WIDTH - 5 ; i++)
                     string[i] = '=';
                  string[i] = '\0';
                  fprintf(stream, "%s\n", string);


                  for (sum = i = 0 ; i < results->year_count ; i++)
                     {
                     sum += results->refuse_data[i];
                     sprintf(string, "     %4d            %13.3E          %12.3E
   %12.3E",
                              landfill_info->year_opened + 1 + i, sum,
                              (volume_to_mass(results->annual_air_toxics[j][i] *
 FT3_TO_M3,
                                              model_info->air_toxics[j].molecula
r_wt,
                                              DEFAULT_TEMP,
                                              DEFAULT_PRESSURE) / 1000.0),
                              results->annual_air_toxics[j][i]);

                     fprintf(stream, "%s\n", string);
                     }
                  }
               else
                  {
                  sprintf(string, "Selection Results were All Zero");
                  fprintf(stream, "          %s\n\n",
                                 strcenter(string, PAGE_WIDTH - 29));
                  }
               }
        }
}



int print_error(filename)
char *filename;
[
menu_type   menu, key_menu;
```

```
sed_type   sed, key_sed;
int        ret;
char string[30];

   strcenter(strcpy(string, filename), 25);

   menu = menu_Open();

   buildbox(menu, error_color, 8, 27);

   menu_Printf(menu, "@p[1,8]@c[%c]Output Error@c[%c]", error_color | BLINK,
                  error_color);
   menu_Printf(menu, "@p[4,4]Error accesing file");
   menu_Printf(menu, "@p[5,1]@fp[@[25,#]]", string, &string_funcs);

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, error_color, error_color, error_color);
   sed_SetPosition(sed, 8, 27);

   sed_Push(sed);
   sed_Repaint(sed);
   show_key_help(&key_menu, &key_sed, "Any Key = Continue");

   kb_Read();

   destroy_key_help(key_menu, key_sed);
   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);
   return(ret);
}



int print_go(filename)
char *filename;
{
menu_type   menu;
sed_type   sed;
int        ret;
char string[30];

   strcenter(strcpy(string, filename), 25);

   menu = menu_Open();

   buildbox(menu, info_color, 8, 27);
   menu_Printf(menu, "@p[1,10]Writing");
   menu_Printf(menu, "@p[4,6]Writing to file");
   menu_Printf(menu, "@p[5,1]@fp[@[25,#]]", string, &string_funcs);

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, info_color, info_color, info_color);
   sed_SetPosition(sed, 8, 27);

   sed_Repaint(sed);

   return(0);
}
```



```
int blank_string(string)
char *string;
{
int i;

   for (i = 0 ; i < PAGE_WIDTH ; i++) string[i] = ' ';
   return(0);
}
```

```
/* ========================================================================== */
/*                                  STUDNAME.C                                */
/* ========================================================================== */
/*  This file contains the code for displaying the study name editor.         */
/* ========================================================================== */
/*                            Created under Contract                          */
/*                                      to                                    */
/*                              EPA/AEERL/GECD/EMB                            */
/*                                                                            */
/*                                      by                                    */
/*                                                                            */
/*                              Radian Corporation                            */
/*                           Research Triangle Park, NC                       */
/* ========================================================================== */
/* ========================================================================== */
/*                              Revision History                              */
/* ========================================================================== */
/*    07/03/90   rlb       Initial Implementation                             */
/* ========================================================================== */

#include <dos.h>
#include <stdio.h>
#include <cscape.h>
#include <string.h>
#include <scancode.h>
#include "landfill.h"
#include "landuser.h"
#include "helpasgn.h"
#include "colors.h"

int get_study_name(study_name, type)
char *study_name;
int type;
{
unsigned count;
char (*buffer)[13];
menu_type   menu, key_menu;
sed_type    sed, key_sed;
int         ret;

   reset_insert_status();

   menu = menu_Open();

   buildbox(menu, menu_color, 7, 39);

   if (type == 2)
      menu_Printf(menu, "@p[1,8]Specify Old Study Name");
   else
      menu_Printf(menu, "@p[1,8]Specify New Study Name");

   menu_Printf(menu, "@p[4,15]@f[@[8,#]]",
     study_name, &dosstr_funcs);

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, menu_color, menu_color, field_color);
   sed_SetPosition(sed, 9, 20);

   sed_Push(sed);
   sed_Repaint(sed);
   show_key_help(&key_menu, &key_sed, studname_help_string);

   do
```



```
      {
      ret = sed_Go(sed);

      switch (ret)
         {
         case FN1:
            compile_help(GET_STUDY_NAME);
            sed_SetBaton(sed, -1);
            break;

         case FN2:
            if (count = compile_file_list(&buffer))
               {
               if (count = file_selector(buffer, count))
                  {
                  strcpy(study_name, buffer[count - 1]);
                  sed_RepaintFields(sed);
                  }
               }
            else
               no_study_found();

            break;
         }
      }
   while ((ret > sed_GetFieldCount(sed)) ||
               ((strlen(study_name) == 0) && (ret)));

   destroy_key_help(key_menu, key_sed);
   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);

   return(ret);
}
```

```
/* ========================================================================== */
/*                                  SUMMARY.C                                 */
/* ========================================================================== */
/*  This file contains the code for displaying the executive summary of       */
/*  the calculations on screen.                                               */
/* ========================================================================== */
/*                             Created under Contract                         */
/*                                       to                                   */
/*                               EPA/AEERL/GECD/EMB                           */
/*                                                                            */
/*                                       by                                   */
/*                                                                            */
/*                                Radian Corporation                          */
/*                             Research Triangle Park, NC                     */
/* ========================================================================== */
/* ========================================================================== */
/*                                Revision History                            */
/* ========================================================================== */
/*    07/03/90    rlb        Initial Implementation                           */
/* ========================================================================== */

#include <stdio.h>
#include <cscape.h>
#include <scancode.h>
#include "landfill.h"
#include "landuser.h"
#include "helpasgn.h"
#include "defaults.h"
#include "constant.h"
#include "colors.h"

int display_summary(landfill_info, data)
struct model_results *data;
struct landfill_parms *landfill_info;
{
menu_type   menu, key_menu;
sed_type    sed, key_sed;
int         ret, i, max_index, curr_index;
double      old_rate;
char format[80];

extern double mass_to_volume(double, double, double, double);
extern double volume_to_mass(double, double, double, double);

   curr_index = landfill_info->current_year
                          - landfill_info->year_opened - 1;

   max_index = curr_index;

   old_rate = data->annual_ch4[0];
   for (i = landfill_info->year_opened;
      i < (landfill_info->year_closed + config.years_passed_closure) ; i++)
         {
         if (data->annual_ch4[i - landfill_info->year_opened] > old_rate)
            {
            max_index = i - landfill_info->year_opened;
            old_rate = data->annual_ch4[i - landfill_info->year_opened];
            }
         }

   menu = menu_Open();

   if (menu == NULL)
      show_mem_error("","");

   menu_Printf(menu, "@p[0,0]@c[%c]+@[78,-]+", info_color);
```



```
   menu_Printf(menu, "@p[1,0]¦");
   menu_Printf(menu, "@p[1,31]Calculation Summary");
   menu_Printf(menu, "@p[1,79]¦");
   menu_Printf(menu, "@p[2,0]¦@[39,-]-@[38,-]¦");
   menu_Printf(menu, "@p[3,0]¦");
   menu_Printf(menu, "@p[3,40]¦");
   menu_Printf(menu, "@p[3,79]¦");
   menu_Printf(menu, "@p[4,0]¦");
   menu_Printf(menu, "@p[4,40]¦");
   menu_Printf(menu, "@p[4,79]¦");
   menu_Printf(menu, "@p[5,0]¦@[39,-]+@[38,-]¦");
   menu_Printf(menu, "@p[6,0]¦");
   menu_Printf(menu, "@p[6,40]¦");
   menu_Printf(menu, "@p[6,79]¦");
   menu_Printf(menu, "@p[7,0]¦ Methane");
   menu_Printf(menu, "@p[7,17]:");
   menu_Printf(menu, "@p[7,34]Mg/yr ¦ Methane");
   menu_Printf(menu, "@p[7,57]:");
   menu_Printf(menu, "@p[7,73]Mg/yr ¦");
   menu_Printf(menu, "@p[8,0]¦");
   menu_Printf(menu, "@p[8,34]Cft/yr¦");
   menu_Printf(menu, "@p[8,73]Cft/yr¦");
   menu_Printf(menu, "@p[9,0]¦");
   menu_Printf(menu, "@p[9,40]¦");
   menu_Printf(menu, "@p[9,79]¦");
   menu_Printf(menu, "@p[10,0]¦ Carbon Dioxide :");
   menu_Printf(menu, "@p[10,34]Mg/yr ¦ Carbon Dioxide :");
   menu_Printf(menu, "@p[10,73]Mg/yr ¦");
   menu_Printf(menu, "@p[11,0]¦");
   menu_Printf(menu, "@p[11,34]Cft/yr¦");
   menu_Printf(menu, "@p[11,73]Cft/yr¦");
   menu_Printf(menu, "@p[12,0]¦");
   menu_Printf(menu, "@p[12,40]¦");
   menu_Printf(menu, "@p[12,79]¦");
   menu_Printf(menu, "@p[13,0]¦ NMOC");
   menu_Printf(menu, "@p[13,17]:");
   menu_Printf(menu, "@p[13,34]Mg/yr ¦ NMOC");
   menu_Printf(menu, "@p[13,57]:");
   menu_Printf(menu, "@p[13,73]Mg/yr ¦");
   menu_Printf(menu, "@p[14,0]¦");
   menu_Printf(menu, "@p[14,34]Cft/yr¦");
   menu_Printf(menu, "@p[14,73]Cft/yr¦");
   menu_Printf(menu, "@p[15,0]¦");
   menu_Printf(menu, "@p[15,40]¦");
   menu_Printf(menu, "@p[15,79]¦");
   menu_Printf(menu, "@p[16,0]+@[39,-]-@[38,-]+");

   sprintf(format, "@p[4,1]     Current Year Results : %4d",
          landfill_info->current_year);
   menu_Printf(menu, format);

   sprintf(format, "@p[4,41]     Maximum Year Results : %4d",
         landfill_info->year_opened + max_index + 1);
   menu_Printf(menu, format);

   sprintf(format, "@p[7,19]%13.3E",
          (volume_to_mass(data->annual_ch4[curr_index] * FT3_TO_M3, 16.04,
                          DEFAULT_TEMP, DEFAULT_PRESSURE) / 1000.0));
   menu_Printf(menu, format);

   sprintf(format, "@p[8,19]%13.3E", data->annual_ch4[curr_index]);
   menu_Printf(menu, format);

   sprintf(format, "@p[10,19]%13.3E",
            (volume_to_mass(data->annual_co2[curr_index] * FT3_TO_M3,
                            44.01, DEFAULT_TEMP, DEFAULT_PRESSURE) / 1000.0));
```

```
   menu_Printf(menu, format);

   sprintf(format, "@p[11,19]%13.3E", data->annual_co2[curr_index]);
   menu_Printf(menu, format);

   sprintf(format, "@p[13,19]%13.3E", data->annual_nmoc[curr_index]);
   menu_Printf(menu, format);

   sprintf(format, "@p[14,19]%13.3E",
      (mass_to_volume(data->annual_nmoc[curr_index] * 1000.0,
                      DEFAULT_NMOC_MOLE_WT, DEFAULT_TEMP,
                      DEFAULT_PRESSURE) / FT3_TO_M3));
   menu_Printf(menu, format);

   sprintf(format, "@p[7,58]%13.3E",
          (volume_to_mass(data->annual_ch4[max_index] * FT3_TO_M3, 16.04,
                          DEFAULT_TEMP, DEFAULT_PRESSURE) / 1000.0));
   menu_Printf(menu, format);

   sprintf(format, "@p[8,58]%13.3E", data->annual_ch4[max_index]);
   menu_Printf(menu, format);

   sprintf(format, "@p[10,58]%13.3E",
          (volume_to_mass(data->annual_co2[max_index] * FT3_TO_M3,
                          44.01, DEFAULT_TEMP, DEFAULT_PRESSURE) / 1000.0));
   menu_Printf(menu, format);

   sprintf(format, "@p[11,58]%13.3E", data->annual_co2[max_index]);
   menu_Printf(menu, format);

   sprintf(format, "@p[13,58]%13.3E", data->annual_nmoc[max_index]);
   menu_Printf(menu, format);

   sprintf(format, "@p[14,58]%13.3E",
      (mass_to_volume(data->annual_nmoc[max_index] * 1000.0,
                      DEFAULT_NMOC_MOLE_WT, DEFAULT_TEMP,
                      DEFAULT_PRESSURE) / FT3_TO_M3));
   menu_Printf(menu, format);

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, info_color, info_color, info_color);
   sed_SetPosition(sed, 4, 0);

   sed_Push(sed);
   sed_Repaint(sed);
   show_key_help(&key_menu, &key_sed, any_key_string);

   kb_Read();

   destroy_key_help(key_menu, key_sed);
   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);
   return(ret);
}
```

```
/* ========================================================================== */
/*                                TBLEMENU.C                                  */
/* ========================================================================== */
/*  This file contains the code for displaying the tabular output device      */
/*  menu.                                                                     */
/* ========================================================================== */
/*                          Created under Contract                            */
/*                                    to                                      */
/*                            EPA/AEERL/GECD/EMB                              */
/*                                                                            */
/*                                    by                                      */
/*                                                                            */
/*                           Radian Corporation                               */
/*                        Research Triangle Park, NC                          */
/* ========================================================================== */
/* ========================================================================== */
/*                           Revision History                                 */
/* ========================================================================== */
/*    07/03/90   rlb       Initial Implementation                             */
/* ========================================================================== */

#include <stdio.h>
#include <cscape.h>
#include <scancode.h>
#include "helpasgn.h"
#include "colors.h"

int tabular_report_menu(choice)
int choice;
{
menu_type   menu, key_menu;
sed_type    sed, key_sed;
int         ret;

   menu = menu_Open();

   buildbox(menu, menu_color, 11, 29);
   menu_Printf(menu, "@p[1,5]Tabular Report Menu");
   menu_Printf(menu, "@p[4,1]@f[     Output to Screen      ]",
     NULL, &menu_funcs);
   menu_Printf(menu, "@p[5,1]@f[     Output to Printer     ]",
     NULL, &menu_funcs);
   menu_Printf(menu, "@p[6,1]@f[      Output to File       ]",
     NULL, &menu_funcs);
   menu_Printf(menu, "@p[8,1]@f[       Previous Menu       ]",
     NULL, &menu_funcs);

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, menu_color, menu_color, field_color);
   sed_SetPosition(sed, 7, 25);

   sed_Push(sed);

   sed_Repaint(sed);
   show_key_help(&key_menu, &key_sed, generic_menu_help_string);

   sed_GotoField(sed, choice);

   do
      {
      ret = sed_Go(sed);

      if (ret == FN1)
```



```
         {
         compile_help(TABULAR_REPORT_MENU);
         sed_SetBaton(sed, -1);
         }
      }
   while (ret > sed_GetFieldCount(sed));

   destroy_key_help(key_menu, key_sed);
   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);
   return(ret);
}
```

```
/* ========================================================================== */
/*                                VFERROR.C                                   */
/* ========================================================================== */
/*  This file contains the code for displaying verification errors.           */
/* ========================================================================== */
/*                           Created under Contract                           */
/*                                    to                                      */
/*                             EPA/AEERL/GECD/EMB                             */
/*                                                                            */
/*                                    by                                      */
/*                                                                            */
/*                             Radian Corporation                             */
/*                          Research Triangle Park, NC                        */
/* ========================================================================== */
/* ========================================================================== */
/*                            Revision History                                */
/* ========================================================================== */
/*    07/03/90   rlb       Initial Implementation                             */
/* ========================================================================== */

#include <stdio.h>
#include <string.h>
#include <cscape.h>
#include "helpasgn.h"
#include "vfmsgs.h"
#include "colors.h"

void verify_error(i)
int i;
{
menu_type   menu, key_menu;
sed_type    sed, key_sed;
int         ret;

   menu = menu_Open();

   buildbox(menu, error_color, 9, 40);

   menu_Printf(menu, "@p[1,12]@c[%c]Validation Error@c[%c]",
                     error_color | BLINK, error_color);
   menu_Printf(menu, "@p[4,2]@fp[@[37,#]]",
     strcenter(error_msg[i].string1, 37), &string_funcs);
   menu_Printf(menu, "@p[5,2]@fp[@[37,#]]",
     strcenter(error_msg[i].string2, 37), &string_funcs);
   menu_Printf(menu, "@p[6,2]@fp[@[37,#]]",
     strcenter(error_msg[i].string3, 37), &string_funcs);

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, error_color, error_color, error_color);
   sed_SetPosition(sed, 8, 20);

   sed_Push(sed);
   sed_Repaint(sed);
   show_key_help(&key_menu, &key_sed, any_key_string);

   kb_Read();

   destroy_key_help(key_menu, key_sed);
   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);
   return;
```



```
}
```

```
/* ========================================================================= */
/*                                WARNINGS.C                                 */
/* ========================================================================= */
/*      This file contains the code for displaying warning popups.           */
/* ========================================================================= */
/*                          Created under Contract                           */
/*                                  to                                       */
/*                            EPA/AEERL/GECD/EMB                             */
/*                                                                           */
/*                                  by                                       */
/*                                                                           */
/*                            Radian Corporation                             */
/*                         Research Triangle Park, NC                        */
/* ========================================================================= */
/* ========================================================================= */
/*                            Revision History                               */
/* ========================================================================= */
/*    07/03/90   rlb       Initial Implementation                            */
/* ========================================================================= */

#include <stdio.h>
#include <cscape.h>
#include <scancode.h>
#include <errno.h>
#include "colors.h"
#include "helpasgn.h"

/* ========================================================================= */
/*  This function displays the warning popup on entry to the k               */
/*  routine.                                                                 */
/* ========================================================================= */

int warn_about_2e()
{
menu_type   menu, key_menu;
sed_type   sed, key_sed;
int       ret;

   menu = menu_Open();

   menu_Printf(menu, "@p[0,0]@c[%c]+@[41,-]+", warning_color);
   menu_Printf(menu, "@p[1,0]¦");
   menu_Printf(menu, "@p[1,42]¦");
   menu_Printf(menu, "@p[2,0]¦  To calculate a k value, you must have  ¦");
   menu_Printf(menu, "@p[3,0]¦  performed the field tests as outlined  ¦");
   menu_Printf(menu, "@p[4,0]¦  in Method 2E.  Press Escape to return  ¦");
   menu_Printf(menu, "@p[5,0]¦  to the previous menu.                  ¦");
   menu_Printf(menu, "@p[6,0]¦");
   menu_Printf(menu, "@p[6,42]¦");
   menu_Printf(menu, "@p[7,0]+@[41,-]+");

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, warning_color, warning_color, warning_color);
   sed_SetPosition(sed, 8, 20);

   sed_Push(sed);
   sed_Repaint(sed);
   show_key_help(&key_menu, &key_sed, abortable_warning_help_string);

   ret = kb_Read();

   destroy_key_help(key_menu, key_sed);
   sed_Pop(sed);
```



```
   menu_Close(menu);
   sed_Close(sed);
   return(ret);
}

/* ========================================================================= */
/*  This function displays a popup warning the user the data entered is      */
/*  not complete enough to allow model calculation.                          */
/* ========================================================================= */

int warn_no_data()
{
menu_type   menu, key_menu;
sed_type   sed, key_sed;
int       ret;

   menu = menu_Open();

   menu_Printf(menu, "@p[0,0]@c[%c]+@[39,-]+", warning_color);
   menu_Printf(menu, "@p[1,0]¦");
   menu_Printf(menu, "@p[1,40]¦");
   menu_Printf(menu, "@p[2,0]¦  The data entered is not sufficient   ¦");
   menu_Printf(menu, "@p[3,0]¦  to allow calculation.  Please enter  ¦");
   menu_Printf(menu, "@p[4,0]¦  landfill parameters for your case    ¦");
   menu_Printf(menu, "@p[5,0]¦  using the \"Operational Data\" item    ¦");
   menu_Printf(menu, "@p[6,0]¦  from the \"Edit Study Data\" menu.     ¦");
   menu_Printf(menu, "@p[7,0]¦");
   menu_Printf(menu, "@p[7,40]¦");
   menu_Printf(menu, "@p[8,0]+@[39,-]+");

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, warning_color, warning_color, warning_color);
   sed_SetPosition(sed, 8, 18);

   sed_Push(sed);
   sed_Repaint(sed);
   show_key_help(&key_menu, &key_sed, any_key_string);

   kb_Read();

   destroy_key_help(key_menu, key_sed);
   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);
   return(ret);
}

/* ========================================================================= */
/*  This function displays the popup warning the user that calculations      */
/*  must be performed before any output may be generated.                    */
/* ========================================================================= */

int warn_no_calc()
{
menu_type   menu, key_menu;
sed_type   sed, key_sed;
int       ret;

   menu = menu_Open();

   menu_Printf(menu, "@p[0,0]@c[%c]+@[39,-]+", warning_color);
   menu_Printf(menu, "@p[1,0]¦");
   menu_Printf(menu, "@p[1,40]¦");
```

```
    menu_Printf(menu, "@p[2,0]|  You have not calculated landfill     |");
    menu_Printf(menu, "@p[3,0]|  emissions.  Please select the        |");
    menu_Printf(menu, "@p[4,0]|  \"Calculate Air Emissions\" item from  |");
    menu_Printf(menu, "@p[5,0]|  the \"Main Menu\".                     |");
    menu_Printf(menu, "@p[6,0]|                                       |");
    menu_Printf(menu, "@p[7,0]|");
    menu_Printf(menu, "@p[7,40]|");
    menu_Printf(menu, "@p[8,0]+@[39,-]+");

    menu_Flush(menu);

    sed = sed_Open(menu);
    sed_SetColors(sed, warning_color, warning_color, warning_color);
    sed_SetPosition(sed, 8, 18);

    sed_Push(sed);
    sed_Repaint(sed);
    show_key_help(&key_menu, &key_sed, any_key_string);

    kb_Read();

    destroy_key_help(key_menu, key_sed);
    sed_Pop(sed);

    menu_Close(menu);
    sed_Close(sed);
    return(ret);
}

/* ========================================================================== */
/*  This function displays the popup which warns the user that a study       */
/*  type must be selected before proceding with the model.                    */
/* ========================================================================== */


void no_study_selected()
{
menu_type   menu, key_menu;
sed_type    sed, key_sed;
int         ret;

    menu = menu_Open();

    menu_Printf(menu, "@p[0,0]@c[%c]+@[37,-]+", warning_color);
    menu_Printf(menu, "@p[1,0]|");
    menu_Printf(menu, "@p[1,38]|");
    menu_Printf(menu, "@p[2,0]|  You are not currently editting a   |");
    menu_Printf(menu, "@p[3,0]|");
    menu_Printf(menu, "@p[3,15]study.");
    menu_Printf(menu, "@p[3,38]|");
    menu_Printf(menu, "@p[4,0]|");
    menu_Printf(menu, "@p[4,38]|");
    menu_Printf(menu, "@p[5,0]|  Please select a study to edit by   |");
    menu_Printf(menu, "@p[6,0]| selecting 'Specify Study' from the  |");
    menu_Printf(menu, "@p[7,0]|");
    menu_Printf(menu, "@p[7,14]main menu.");
    menu_Printf(menu, "@p[7,38]|");
    menu_Printf(menu, "@p[8,0]|");
    menu_Printf(menu, "@p[8,38]|");
    menu_Printf(menu, "@p[9,0]+@[37,-]+");

    menu_Flush(menu);

    sed = sed_Open(menu);
    sed_SetColors(sed, warning_color, warning_color, warning_color);
    sed_SetPosition(sed, 8, 19);
```



```
    sed_Push(sed);
    sed_Repaint(sed);
    show_key_help(&key_menu, &key_sed, any_key_string);

    kb_Read();

    destroy_key_help(key_menu, key_sed);
    sed_Pop(sed);

    menu_Close(menu);
    sed_Close(sed);
    return;
}


/* ========================================================================== */
/*  This function displays a popup warning the user that no studies          */
/*  were found in the working directory (displayed on activation of the      */
/*  function selector).                                                       */
/* ========================================================================== */

void no_study_found()
{
menu_type   menu, key_menu;
sed_type    sed, key_sed;
int         ret;

    menu = menu_Open();

    menu_Printf(menu, "@p[0,0]@c[%c]+@[37,-]+", warning_color);
    menu_Printf(menu, "@p[1,0]|");
    menu_Printf(menu, "@p[1,38]|");
    menu_Printf(menu, "@p[2,0]|     No studies were found in the     |");
    menu_Printf(menu, "@p[3,0]|      current working directory       |");
    menu_Printf(menu, "@p[4,0]|");
    menu_Printf(menu, "@p[4,38]|");
    menu_Printf(menu, "@p[5,0]+@[37,-]+");

    menu_Flush(menu);

    sed = sed_Open(menu);
    sed_SetColors(sed, warning_color, warning_color, warning_color);
    sed_SetPosition(sed, 9, 20);

    sed_Push(sed);
    sed_Repaint(sed);
    show_key_help(&key_menu, &key_sed, any_key_string);

    kb_Read();

    destroy_key_help(key_menu, key_sed);
    sed_Pop(sed);

    menu_Close(menu);
    sed_Close(sed);
    return;
}


/* ========================================================================== */
/*  This function warns the user that the action taken (pressing F2 in       */
/*  the air toxics menu screen) will reset the air toxics list.  It also     */
/*  returns the response from the user which is used to indicate whether     */
/*  the user intends to abort this process or proceed with resetting the     */
/*  list.                                                                     */
```

```
/* ========================================================================== */


int warn_about_reset()
{
menu_type   menu, key_menu;
sed_type    sed, key_sed;
int         ret;
unsigned old_cursor;

   old_cursor = vid_GetCursorType();
   vid_SetCursorType(CURSOR_NONE);

   menu = menu_Open();

   menu_Printf(menu, "@p[0,0]@c[%c]+@[38,-]+", warning_color);
   menu_Printf(menu, "@p[1,0]¦");
   menu_Printf(menu, "@p[1,16]CAUTION");
   menu_Printf(menu, "@p[1,39]¦");
   menu_Printf(menu, "@p[2,0]¦@[38,-]¦");
   menu_Printf(menu, "@p[3,0]¦");
   menu_Printf(menu, "@p[3,39]¦");
   menu_Printf(menu, "@p[4,0]¦ The action you are taking will reset ¦");
   menu_Printf(menu, "@p[5,0]¦  the ENTIRE air toxics list to its   ¦");
   menu_Printf(menu, "@p[6,0]¦");
   menu_Printf(menu, "@p[6,13]default state.");
   menu_Printf(menu, "@p[6,39]¦");
   menu_Printf(menu, "@p[7,0]¦");
   menu_Printf(menu, "@p[7,39]¦");
   menu_Printf(menu, "@p[8,0]+@[38,-]+");

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, warning_color, warning_color, field_color);
   sed_SetPosition(sed, 7, 21);

   sed_Push(sed);
   sed_Repaint(sed);
   show_key_help(&key_menu, &key_sed, abortable_warning_help_string);

   ret = kb_Read();

   destroy_key_help(key_menu, key_sed);
   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);

   vid_SetCursorType(old_cursor);

   return(((ret == ESC) ? 0 : 1));
}



int warn_about_nmoc_default()
{
menu_type   menu, key_menu;
sed_type    sed, key_sed;
int         ret;

   menu = menu_Open();

   menu_Printf(menu, "@p[0,0]@c[%c]+@[38,-]+", warning_color);
   menu_Printf(menu, "@p[1,0]¦");
```



```
   menu_Printf(menu, "@p[1,39]¦");
   menu_Printf(menu, "@p[2,0]¦  You have changed the default        ¦");
   menu_Printf(menu, "@p[3,0]¦  disposal type originally specified  ¦");
   menu_Printf(menu, "@p[4,0]¦  for the landfill.  This directly    ¦");
   menu_Printf(menu, "@p[5,0]¦  influences the selection of the     ¦");
   menu_Printf(menu, "@p[6,0]¦  default NMOC concentration.  Please ¦");
   menu_Printf(menu, "@p[7,0]¦  examine the NMOC value and, if      ¦");
   menu_Printf(menu, "@p[8,0]¦  necessary, reset/change the value.  ¦");
   menu_Printf(menu, "@p[9,0]¦");
   menu_Printf(menu, "@p[9,39]¦");
   menu_Printf(menu, "@p[10,0]+@[38,-]+");

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, warning_color, warning_color, warning_color);
   sed_SetPosition(sed, 7, 20);

   sed_Push(sed);
   sed_Repaint(sed);
   show_key_help(&key_menu, &key_sed, abortable_warning_help_string);

   ret = kb_Read();

   destroy_key_help(key_menu, key_sed);
   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);
   return(ret);
}



/* ========================================================================== */
/*  This function is similar to the one above.  It warns the user of the  */
/*  potential resetting of the operational periods list.                  */
/* ========================================================================== */

int warn_about_period_reset()
{
menu_type   menu, key_menu;
sed_type    sed, key_sed;
int         ret;
unsigned old_cursor;

   old_cursor = vid_GetCursorType();
   vid_SetCursorType(CURSOR_NONE);

   menu = menu_Open();

   menu_Printf(menu, "@p[0,0]@c[%c]+@[38,-]+", warning_color);
   menu_Printf(menu, "@p[1,0]¦");
   menu_Printf(menu, "@p[1,16]CAUTION");
   menu_Printf(menu, "@p[1,39]¦");
   menu_Printf(menu, "@p[2,0]¦@[38,-]¦");
   menu_Printf(menu, "@p[3,0]¦");
   menu_Printf(menu, "@p[3,39]¦");
   menu_Printf(menu, "@p[4,0]¦ The action you are taking will reset ¦");
   menu_Printf(menu, "@p[5,0]¦    the ENTIRE period list to its     ¦");
   menu_Printf(menu, "@p[6,0]¦");
   menu_Printf(menu, "@p[6,13]default state.");
   menu_Printf(menu, "@p[6,39]¦");
   menu_Printf(menu, "@p[7,0]¦");
   menu_Printf(menu, "@p[7,39]¦");
   menu_Printf(menu, "@p[8,0]+@[38,-]+");
```

```
   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, warning_color, warning_color, warning_color);
   sed_SetPosition(sed, 8, 21);

   sed_Push(sed);
   sed_Repaint(sed);
   show_key_help(&key_menu, &key_sed, abortable_warning_help_string);

   ret = kb_Read();

   destroy_key_help(key_menu, key_sed);
   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);

   vid_SetCursorType(old_cursor);

   return(((ret == ESC) ? 0 : 1));
}


/* ========================================================================= */
/*  This function displays an error box indicating that a dos file error    */
/*  has occurred.                                                            */
/* ========================================================================= */

int show_file_error()
{
menu_type   menu, key_menu;
sed_type    sed, key_sed;
int         ret;

   menu = menu_Open();

   buildbox(menu, error_color, 8, 31);

   menu_Printf(menu, "@p[1,11]@c[%c]DOS Error@c[%c]", error_color | BLINK,
               error_color);
   menu_Printf(menu, "@p[4,4]Unable to open specified");
   menu_Printf(menu, "@p[5,13]file");
   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, error_color, error_color, '\x70');
   sed_SetPosition(sed, 8, 23);

   sed_Push(sed);
   sed_Repaint(sed);
   show_key_help(&key_menu, &key_sed, any_key_string);

   kb_Read();

   destroy_key_help(key_menu, key_sed);
   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);
   return(ret);
}


/* ========================================================================= */
```



```
/*  This function displays a popup indicating that a non-catastrophic      */
/*  memory error has occurred.  This type of error only occurs as a         */
/*  result of selecting the calculation option and indicates that memory    */
/*  is not sufficient to calculate all of the parameters desired.           */
/* ========================================================================= */

int show_mem_error2()
{
menu_type   menu, key_menu;
sed_type    sed, key_sed;
int         ret;

   menu = menu_Open();

   buildbox(menu, error_color, 15, 41);

   menu_Printf(menu, "@p[1,13]Memory Error #2");
   menu_Printf(menu, "@p[4,3]You do not have enough free memory");
   menu_Printf(menu, "@p[5,3]to perform the calculation you are");
   menu_Printf(menu, "@p[6,3]requesting.");
   menu_Printf(menu, "@p[8,3]You may either reduce the number of");
   menu_Printf(menu, "@p[9,3]desired periods or the number of");
   menu_Printf(menu, "@p[10,3]air toxics and attempt this step");
   menu_Printf(menu, "@p[11,3]again or exit the program and");
   menu_Printf(menu, "@p[12,3]remove memory resident programs and");
   menu_Printf(menu, "@p[13,3]execute the program again.");

   menu_Flush(menu);

   sed = sed_Open(menu);
   sed_SetColors(sed, error_color, error_color, error_color);
   sed_SetPosition(sed, 5, 19);

   sed_Push(sed);
   sed_Repaint(sed);
   show_key_help(&key_menu, &key_sed, any_key_string);

   kb_Read();

   destroy_key_help(key_menu, key_sed);
   sed_Pop(sed);

   menu_Close(menu);
   sed_Close(sed);

   return(ret);
}


int show_mem_limitation()
{
menu_type   menu, key_menu;
sed_type    sed, key_sed;
int         ret;

   errno = 0;

   menu = menu_Open();

   buildbox(menu, error_color, 15, 41);

   menu_Printf(menu, "@p[1,13]Memory Error #3");
   menu_Printf(menu, "@p[4,3]You do not have enough free memory");
   menu_Printf(menu, "@p[5,3]to execute the landfill program.");
   menu_Printf(menu, "@p[7,3]This system requires a minimum of");
   menu_Printf(menu, "@p[8,3]400 Kbytes of memory.  Please ensure");
```

```
    menu_Printf(menu, "@p[9,3]your system has enough memory to");
    menu_Printf(menu, "@p[10,3]meet this requirement and that you");
    menu_Printf(menu, "@p[11,3]have no memory resident programs");
    menu_Printf(menu, "@p[12,3]limiting the available free memory");

    menu_Flush(menu);

    sed = sed_Open(menu);
    sed_SetColors(sed, error_color, error_color, error_color);
    sed_SetPosition(sed, 5, 19);

    sed_Push(sed);
    sed_Repaint(sed);

    if (errno) exit(0);
    show_key_help(&key_menu, &key_sed, any_key_string);

    kb_Read();

    destroy_key_help(key_menu, key_sed);
    sed_Pop(sed);

    menu_Close(menu);
    sed_Close(sed);

    return(ret);
}



int show_mem_error4()
{
menu_type   menu, key_menu;
sed_type    sed, key_sed;
int         ret;

    menu = menu_Open();

    buildbox(menu, error_color, 11, 41);

    menu_Printf(menu, "@p[1,13]Memory Error #4");
    menu_Printf(menu, "@p[4,3]You do not have enough free memory");
    menu_Printf(menu, "@p[5,3]to display the report interactively.");
    menu_Printf(menu, "@p[7,3]You may, on the other hand, print");
    menu_Printf(menu, "@p[8,3]this information or save the result");
    menu_Printf(menu, "@p[9,3]to a file for later examination.");

    menu_Flush(menu);

    sed = sed_Open(menu);
    sed_SetColors(sed, error_color, error_color, error_color);
    sed_SetPosition(sed, 7, 19);

    if (errno != ENOMEM) sed_Push(sed);
    sed_Repaint(sed);
    show_key_help(&key_menu, &key_sed, any_key_string);

    kb_Read();

    destroy_key_help(key_menu, key_sed);
    if (errno != ENOMEM) sed_Pop(sed);

    menu_Close(menu);
    sed_Close(sed);
```



```
    return(ret);
}



/* ========================================================================= */
/*  This function displays an error message for a catastrophic memory      */
/*  error.                                                                 */
/* ========================================================================= */

void show_mem_error(string1, string2)
char *string1, *string2;
{
    printf("ERROR NUMBER 1 in %s: %s\n", string1, string2);
    printf("Having difficulty allocating enough memory.\n");
    printf("Please remove any memory resident programs and\n");
    printf("attempt to execute.  If that does not work,\n");
    printf("please report the error.\n");
    printf("\nPress any key to return to DOS ...\n");
    getchar();
    exit(-1);
}
```

Appendix C

Windows Version Linker Definition File

```
NAME        LANDWIN
DESCRIPTION 'Landfill Air Emissions Estimation Model'
EXETYPE     WINDOWS
CODE        PRELOAD MOVEABLE
DATA        PRELOAD MOVEABLE MULTIPLE
HEAPSIZE    16384
STACKSIZE   32767
EXPORTS     WNDPROC
            FPNUMBEREDITPROC
            INTNUMBEREDITPROC
            CONTAINERWNDPROC
```

Appendix D

Windows Version Resource File

```
#include "\bc45\include\windows.h"
#include "landres.h"


LANDWIN_MENU MENU
{
 POPUP "&File"
 {
  MENUITEM "&New", LAND_MENU_NEW
  MENUITEM "&Open", LAND_MENU_OPEN
  MENUITEM SEPARATOR
  MENUITEM "&Save", LAND_MENU_SAVE, GRAYED
  MENUITEM "Save &As", LAND_MENU_SAVEAS, GRAYED
  MENUITEM SEPARATOR
  MENUITEM "&Print", LAND_MENU_PRINT
  MENUITEM SEPARATOR
  MENUITEM "&Options ...", LAND_MENU_OPTIONS
  MENUITEM SEPARATOR
  MENUITEM "E&xit\tAlt-F4", LAND_MENU_EXIT
 }

 POPUP "&Utilities"
 {
  MENUITEM "Unit &Conversions", LAND_MENU_CONVERT
  MENUITEM "&Refuse Estimator", LAND_MENU_ESTIMATE
 }

 POPUP "&Help"
 {
  MENUITEM "&Index\tF1", LAND_MENU_HELP_INDEX
  MENUITEM SEPARATOR
  MENUITEM "&About...", LAND_MENU_ABOUT
 }

}


LANDWIN_OPER MENU
{
 POPUP "&File"
 {
  MENUITEM "&New", LAND_MENU_NEW
  MENUITEM "&Open", LAND_MENU_OPEN
  MENUITEM SEPARATOR
  MENUITEM "&Save", LAND_MENU_SAVE
  MENUITEM "Save &As", LAND_MENU_SAVEAS
  MENUITEM SEPARATOR
  MENUITEM "&Print", LAND_MENU_PRINT
  MENUITEM SEPARATOR
  MENUITEM "&Options ...", LAND_MENU_OPTIONS
  MENUITEM SEPARATOR
  MENUITEM "E&xit\tAlt-F4", LAND_MENU_EXIT
 }

 POPUP "&Edit"
 {
  MENUITEM "Cu&t\tShift-Del", LAND_MENU_CUT
  MENUITEM "&Copy\tCtrl-Ins", LAND_MENU_COPY
  MENUITEM "&Paste\tShift-Ins", LAND_MENU_PASTE
 }

 POPUP "&Defaults"
 {
  MENUITEM "&CAA", LAND_MENU_REGULATORY, CHECKED
  MENUITEM "&AP-42", LAND_MENU_AP42
  MENUITEM "&User Specific", LAND_MENU_SPECIFIC
```



```
  MENUITEM SEPARATOR
  MENUITEM "Current &Parameters ...", LAND_MENU_CURRENT
 }

 POPUP "&Parameters"
 {
  POPUP "&Generation Rate (k)"
  {
   MENUITEM "Default &CAA", LAND_MENU_DEFAULT_REG_K, CHECKED
   MENUITEM "Default &AP42", LAND_MENU_DEFAULT_AP42_K, GRAYED
   MENUITEM "&User Specified ...", LAND_MENU_USER_K, GRAYED
   MENUITEM "Ca&lculated ...", LAND_MENU_CALC_K, GRAYED
  }

  POPUP "&Generation Potential (lo)"
  {
   MENUITEM "&Default CAA", LAND_MENU_DEFAULT_REG_L, CHECKED
   MENUITEM "Default &AP-42", LAND_MENU_DEFAULT_AP42_L, GRAYED
   MENUITEM "&User Specified ...", LAND_MENU_USER_L, GRAYED
  }

  MENUITEM SEPARATOR
  POPUP "&Landfill Type"
  {
   MENUITEM "&No Codisposal", LAND_MENU_NOCO
   MENUITEM "&Codisposal", LAND_MENU_CODIS
  }

  MENUITEM SEPARATOR
  MENUITEM "&Air Pollutants ...", LAND_MENU_AIRTOXICS
  MENUITEM SEPARATOR
  MENUITEM "Closure &Year ...", LAND_MENU_CLOSURE
 }

 POPUP "&Reports"
 {
  MENUITEM "&Text ...", LAND_MENU_TEXT
  MENUITEM "&Graphics ...", LAND_MENU_GRAPHICS
 }

 POPUP "&Utilities"
 {
  MENUITEM "Unit &Conversions", LAND_MENU_CONVERT
  MENUITEM "&Refuse Estimator", LAND_MENU_ESTIMATE
 }

 POPUP "&Window"
 {
  MENUITEM "Tile &Vertical", LAND_MENU_TILE_VERTICAL
  MENUITEM "Tile &Horizontal", LAND_MENU_TILE_HORIZONTAL
  MENUITEM "&Cascade", LAND_MENU_CASCADE
  MENUITEM "&Arrange Icons", LAND_MENU_ICONARRANGE
 }

 POPUP "&Help"
 {
  MENUITEM "&Index\tF1", LAND_MENU_HELP_INDEX
  MENUITEM SEPARATOR
  MENUITEM "&About...", LAND_MENU_ABOUT
 }

}

ICON_1 ICON "landwin.ico"

K_DIALOG DIALOG 79, 107, 194, 89
```

```
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CLASS "bordlg"
CAPTION "Methane Generation Rate (k)"
FONT 8, "MS Sans Serif"
{
 DEFPUSHBUTTON "OK", IDC_OK, 12, 66, 50, 14
 PUSHBUTTON "Cancel", IDC_CANCEL, 72, 66, 50, 14
 PUSHBUTTON "Help", IDC_HELP, 132, 66, 50, 14
 CONTROL "", -1, "BorShade", BSS_HDIP | BSS_LEFT | WS_CHILD | WS_VISIBLE, 0, 53,
 194, 2
 CONTROL "", -1, "BorShade", BSS_GROUP | BSS_CAPTION | BSS_LEFT | WS_CHILD | WS_
VISIBLE, 12, 10, 169, 31
 EDITTEXT IDC_VALUEEDIT, 45, 20, 102, 12
}

L_DIALOG DIALOG 109, 28, 197, 85
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CLASS "bordlg"
CAPTION "Methane Generation Potential"
FONT 8, "MS Sans Serif"
{
 DEFPUSHBUTTON "OK", IDC_OK, 13, 63, 50, 14
 PUSHBUTTON "Cancel", IDC_CANCEL, 73, 63, 50, 14
 PUSHBUTTON "Help", IDC_HELP, 133, 63, 50, 14
 CONTROL "", -1, "BorShade", BSS_HDIP | BSS_LEFT | WS_CHILD | WS_VISIBLE, 1, 50,
 194, 2
 CONTROL "", -1, "BorShade", BSS_GROUP | BSS_CAPTION | BSS_LEFT | WS_CHILD | WS_
VISIBLE, 13, 7, 169, 31
 EDITTEXT IDC_VALUEEDIT, 46, 17, 102, 12
}


ABOUT_DIALOG DIALOG 14, 32, 250, 145
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CLASS "bordlg"
CAPTION "About LAEEMS"
FONT 8, "MS Sans Serif"
{
 DEFPUSHBUTTON "OK", IDC_OK, 100, 125, 50, 14
 CONTROL "Landfill Air Emissions Estimation Model", -1, "BorStatic", SS_CENTER |
 WS_CHILD | WS_VISIBLE | WS_GROUP, 44, 11, 161, 8
 CONTROL "", -1, "BorShade", BSS_GROUP | BSS_CAPTION | BSS_LEFT | WS_CHILD | WS_
VISIBLE, 10, 6, 229, 31
 CONTROL "", -1, "BorShade", BSS_GROUP | BSS_CAPTION | BSS_LEFT | WS_CHILD | WS_
VISIBLE, 10, 42, 229, 69
 CONTROL "", -1, "BorShade", BSS_HDIP | BSS_LEFT | WS_CHILD | WS_VISIBLE, 0, 117
, 250, 2
 CONTROL "Version 1.0", -1, "BorStatic", SS_CENTER | WS_CHILD | WS_VISIBLE | WS_
GROUP, 91, 23, 65, 8
 CONTROL "Prepared under contract to ", -1, "BorStatic", SS_CENTER | WS_CHILD |
WS_VISIBLE | WS_GROUP, 48, 49, 153, 10
 CONTROL "Radian Corporation", -1, "BorStatic", SS_CENTER | WS_CHILD | WS_VISIBL
E | WS_GROUP, 64, 83, 122, 8
 CONTROL "THE UNITED STATES ENVIRONMENTAL PROTECTION AGENCY", -1, "BorStatic", S
S_CENTER | WS_CHILD | WS_VISIBLE | WS_GROUP, 26, 60, 198, 8
 CONTROL "by ", -1, "BorStatic", SS_CENTER | WS_CHILD | WS_VISIBLE | WS_GROUP, 5
6, 71, 138, 8
 CONTROL "Research Triangle Park, North Carolina", -1, "BorStatic", SS_CENTER |
WS_CHILD | WS_VISIBLE | WS_GROUP, 47, 94, 154, 8
 ICON "EPA_LOGO", -1, 210, 12, 18, 20
 ICON "EPA_LOGO", -1, 22, 12, 18, 20
}


OPTIONS_DIALOG DIALOG 34, 23, 194, 178
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
```



```
CLASS "bordlg"
CAPTION "Set Model Options"
FONT 8, "MS Sans Serif"
{
 DEFPUSHBUTTON "&Save", IDC_OK, 12, 154, 50, 14
 PUSHBUTTON "&Cancel", IDC_CANCEL, 72, 154, 50, 14
 PUSHBUTTON "&Help", IDC_HELP, 132, 154, 50, 14
 CONTROL "", -1, "BorShade", BSS_GROUP | BSS_CAPTION | BSS_LEFT | WS_CHILD | WS_
VISIBLE, 10, 9, 174, 122
 CONTROL "", -1, "BorShade", BSS_HDIP | BSS_LEFT | WS_CHILD | WS_VISIBLE, 0, 143
, 194, 2
 CONTROL "Default Mode", -1, "BorShade", BSS_GROUP | BSS_CAPTION | BSS_LEFT | WS
_CHILD | WS_VISIBLE, 25, 20, 144, 66
 CONTROL "CAA Mode", IDC_RADIOBUTTON1, "BorRadio", BS_AUTORADIOBUTTON | WS_CHILD
 | WS_VISIBLE | WS_TABSTOP, 62, 36, 70, 10
 CONTROL "AP-42 Mode", IDC_RADIOBUTTON2, "BorRadio", BS_AUTORADIOBUTTON | WS_CHI
LD | WS_VISIBLE | WS_TABSTOP, 62, 52, 70, 10
 CONTROL "User Specific", IDC_RADIOBUTTON3, "BorRadio", BS_AUTORADIOBUTTON | WS_
CHILD | WS_VISIBLE | WS_TABSTOP, 62, 68, 70, 10
 CONTROL "Calculate Years Past Closure ", -1, "BorStatic", SS_LEFT | WS_CHILD |
WS_VISIBLE | WS_GROUP, 34, 105, 100, 8
 EDITTEXT IDC_VALUEEDIT, 140, 103, 17, 11, ES_RIGHT | WS_BORDER | WS_TABSTOP
}

CALCULATE_DIALOG DIALOG 14, 25, 229, 255
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CLASS "bordlg"
CAPTION "Method 2E Calculator"
FONT 8, "MS Sans Serif"
{
 LTEXT "Average Well &Depth (m)", -1, 12, 14, 115, 10
 EDITTEXT IDC_WELLDEPTH, 146, 14, 68, 11, WS_BORDER | WS_GROUP | WS_TABSTOP
 LTEXT "Average Stabilized Radius of &Influence (m)", -1, 12, 31, 115, 19
 EDITTEXT IDC_ROI, 146, 35, 68, 11, WS_BORDER | WS_GROUP | WS_TABSTOP
 LTEXT "&Refuse Density (Mg/m^3)", -1, 12, 58, 115, 9
 EDITTEXT IDC_DENSITY, 146, 58, 68, 11, WS_BORDER | WS_GROUP | WS_TABSTOP
 LTEXT "&Fraction of Decomposable Refuse", -1, 12, 80, 115, 9
 EDITTEXT IDC_FRACTION, 146, 79, 68, 11, WS_BORDER | WS_GROUP | WS_TABSTOP
 LTEXT "Methane Generation Potential (&L0)  (m^3/Mg)", -1, 12, 98, 115, 19
 EDITTEXT IDC_L0, 146, 101, 68, 11, WS_BORDER | WS_GROUP | WS_TABSTOP
 LTEXT "Average &Stabilized Flow Rate per Well (m^3/min)", -1, 12, 123, 115, 19
 EDITTEXT IDC_FLOWRATE, 146, 126, 68, 11, WS_BORDER | WS_GROUP | WS_TABSTOP
 LTEXT "Average &Age of Refuse (years)", -1, 12, 150, 115, 8
 EDITTEXT IDC_AGE, 146, 149, 68, 11, WS_BORDER | WS_GROUP | WS_TABSTOP
 LTEXT "&k (1/yr)", -1, 84, 200, 26, 8
 PUSHBUTTON "Calculate --->", IDC_CALCULATE, 27, 197, 50, 14, WS_GROUP | WS_TABS
TOP
 DEFPUSHBUTTON "&OK", IDC_OK, 29, 236, 50, 14, BS_DEFPUSHBUTTON | WS_GROUP | WS_
TABSTOP
 PUSHBUTTON "&Cancel", IDC_CANCEL, 89, 236, 50, 14, WS_GROUP | WS_TABSTOP
 PUSHBUTTON "&Help", IDC_HELP, 149, 236, 50, 14, WS_GROUP | WS_TABSTOP
 CONTROL "", -1, "BorShade", BSS_HDIP | BSS_LEFT | WS_CHILD | WS_VISIBLE, 0, 180
, 229, 2
 CONTROL "", -1, "BorShade", BSS_HDIP | BSS_LEFT | WS_CHILD | WS_VISIBLE, 0, 225
, 229, 2
 CONTROL "", -1, "BorShade", BSS_GROUP | BSS_CAPTION | BSS_LEFT | WS_CHILD | WS_
VISIBLE, 117, 196, 89, 16
 LTEXT "", IDC_VALUETEXT, 121, 200, 82, 8
 CONTROL "", -1, "BorShade", BSS_GROUP | BSS_CAPTION | BSS_LEFT | WS_CHILD | WS_
VISIBLE, 7, 10, 214, 160
 CONTROL "", -1, "BorShade", BSS_GROUP | BSS_CAPTION | BSS_LEFT | WS_CHILD | WS_
VISIBLE, 14, 193, 202, 22
}

AIR_TOXICS_DIALOG DIALOG 6, 17, 230, 227
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
```

```
CLASS "bordlg"
CAPTION "Air Pollutant Compound Parameters"
FONT 8, "MS Sans Serif"
{
 LTEXT "Methane Percentage (%)", -1, 22, 27, 105, 8
 EDITTEXT IDC_CH4PERCENT, 139, 25, 68, 11, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_
BORDER | WS_GROUP | WS_TABSTOP
 LTEXT "Carbon Dioxide Percentage (%)", -1, 22, 48, 105, 8
 CONTROL "AP-42", IDC_RADIOAP42, "BorRadio", BS_AUTORADIOBUTTON | WS_CHILD | WS_
VISIBLE | WS_TABSTOP, 44, 103, 91, 10
 CONTROL "CAA", IDC_RADIOREG, "BorRadio", BS_AUTORADIOBUTTON | WS_CHILD | WS_VIS
IBLE | WS_TABSTOP, 44, 86, 140, 10
 CONTROL "User Specified:", IDC_RADIOUSER, "BorRadio", BS_AUTORADIOBUTTON | WS_C
HILD | WS_VISIBLE | WS_TABSTOP, 44, 120, 64, 10
 EDITTEXT IDC_NMOCCONC, 119, 119, 68, 11, ES_LEFT | WS_CHILD | WS_VISIBLE | WS_B
ORDER | WS_GROUP | WS_TABSTOP
 PUSHBUTTON "Edit Air Pollutants ...", IDC_ATBUTTON, 73, 148, 83, 14
 DEFPUSHBUTTON "OK", IDC_OK, 30, 202, 50, 14
 PUSHBUTTON "Cancel", IDC_CANCEL, 90, 202, 50, 14
 PUSHBUTTON "Help", IDC_HELP, 150, 202, 50, 14
 CONTROL "", -1, "BorShade", BSS_GROUP | BSS_CAPTION | BSS_LEFT | WS_CHILD | WS_
VISIBLE, 11, 14, 209, 158
 CONTROL "", -1, "BorShade", BSS_HDIP | BSS_LEFT | WS_CHILD | WS_VISIBLE, 0, 189
, 230, 2
 CONTROL "NMOC Concentration (ppmV)", -1, "BorShade", BSS_GROUP | BSS_CAPTION |
BSS_CENTER | WS_CHILD | WS_VISIBLE | WS_GROUP, 24, 72, 184, 64
 CONTROL "", -1, "BorShade", BSS_GROUP | BSS_CAPTION | BSS_LEFT | WS_CHILD | WS_
VISIBLE, 139, 45, 68, 14
 CONTROL "", IDC_CO2PERCENT, "BorStatic", SS_LEFT | WS_CHILD | WS_VISIBLE | WS_G
ROUP, 141, 48, 43, 8
}

DIALOG_1 DIALOG 54, 48, 194, 119
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CAPTION "About Graphics Display Engine"
FONT 8, "MS Sans Serif"
{
 DEFPUSHBUTTON "OK", IDC_OK, 72, 96, 50, 14
 ICON "LANDGRAPH_ICON", -1, 19, 9, 18, 20
 LTEXT "Graphics Display Engine", -1, 63, 10, 105, 8
 LTEXT "Version 1.0", -1, 63, 18, 60, 8
 LTEXT "Written as a companion facility for programs", -1, 21, 37, 153, 8
 LTEXT "which display graphical data in the Windows", -1, 21, 45, 153, 8
 LTEXT "operating environment.", -1, 21, 53, 153, 8
 CTEXT "Written by Robert Bass of", -1, 21, 68, 152, 9
 CTEXT "RADIAN CORPORATION (RTP).", -1, 21, 77, 152, 9
}

STRINGTABLE
BEGIN
  IDS_WMFFILTER  "Metafiles (*.WMF)|*.WMF|"
  IDS_TEXTFILTER "Text Files (*.TXT)|*.TXT|"
END


DIALOG_PRINT DIALOG 30, 54, 207, 111
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CLASS "bordlg"
CAPTION "Print a Graph"
FONT 8, "MS Sans Serif"
{
 DEFPUSHBUTTON "Print", IDC_PRINT, 147, 6, 50, 14
 PUSHBUTTON "Cancel", IDC_CANCEL, 147, 24, 50, 14
 LTEXT "Available Printers:", -1, 7, 6, 121, 8
 LISTBOX IDC_LISTBOX1, 7, 20, 121, 87, LBS_NOTIFY | WS_BORDER | WS_BORDER | WS_V
SCROLL
```



```
 PUSHBUTTON "Setup ...", IDC_SETUP, 147, 87, 50, 14
}


OPERTOOLDLG DIALOG 0, 0, 297, 53
STYLE WS_CHILD | WS_VISIBLE
FONT 8, "Arial"
{
 CONTROL "Year Opened", -1, "BorStatic", SS_LEFT | WS_CHILD | WS_VISIBLE | WS_GR
OUP, 4, 5, 46, 8
 EDITTEXT IDC_OPENYEAR, 53, 3, 18, 12
 CONTROL "Current Year", -1, "BorStatic", SS_LEFT | WS_CHILD | WS_VISIBLE | WS_G
ROUP, 86, 5, 46, 8
 EDITTEXT IDC_CURRENTYEAR, 134, 3, 18, 12
 CONTROL "Capacity", -1, "BorStatic", SS_LEFT | WS_CHILD | WS_VISIBLE | WS_GROUP
, 168, 5, 31, 8
 EDITTEXT IDC_CAPACITY, 201, 3, 83, 12
 CONTROL "Waste Value", -1, "BorStatic", SS_LEFT | WS_CHILD | WS_VISIBLE | WS_GR
OUP, 4, 30, 44, 8
 EDITTEXT IDC_VALUEEDIT, 51, 28, 83, 12
 CONTROL "as", -1, "BorStatic", SS_LEFT | WS_CHILD | WS_VISIBLE | WS_GROUP, 141,
 30, 16, 8
 CONTROL "Refuse in Place (Mg)", IDC_RADIOBUTTON1, "BorRadio", BS_AUTORADIOBUTTO
N | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 157, 23, 86, 10
 CONTROL "Acceptance Rate (Mg/Yr)", IDC_RADIOBUTTON2, "BorRadio", BS_AUTORADIOBU
TTON | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 157, 38, 97, 10
 CONTROL "", -1, "BorShade", BSS_GROUP | BSS_CAPTION | BSS_LEFT | WS_CHILD | WS_
VISIBLE, -1, 0, 293, 18
 CONTROL "", -1, "BorShade", BSS_GROUP | BSS_CAPTION | BSS_LEFT | WS_CHILD | WS_
VISIBLE, -1, 18, 293, 35
 CONTROL "", -1, "BorShade", BSS_VDIP | BSS_LEFT | WS_CHILD | WS_VISIBLE, 78, 0,
 4, 18
 CONTROL "", -1, "BorShade", BSS_VDIP | BSS_LEFT | WS_CHILD | WS_VISIBLE, 159, 0
, 4, 18
}


EPA_LOGO ICON "about.ico"
LANDOP_ICON ICON "landfac.ico"



STRINGTABLE
{
 DEFAULT_CH4, "50.0"
 DEFAULT_CO2, "50.0"
 DEFAULT_REGULATORY_L0, "169.90108"
 DEFAULT_REGULATORY_K, "0.05"
 DEFAULT_REGULATORY_NMOC, "4000.0"
 DEFAULT_AP42_L0, "124.9056"
 DEFAULT_AP42_K, "0.04"
 DEFAULT_AP42_NMOC_CODIS, "2420"
 DEFAULT_AP42_NMOC_NOCO, "595"
 DEFAULT_WELL_DEPTH, "0.0"
 DEFAULT_INFLUENCE_RADIUS, "0.0"
 DEFAULT_REFUSE_DENSITY, "0.636"
 DEFAULT_DECOMP_FRACTION, "1.0"
 DEFAULT_FLOW_RATE, "0.0"
 DEFAULT_REFUSE_AGE, "0.0"
 DEFAULT_TEMP, "293.0"
 DEFAULT_PRESSURE, "1.01325E5"
 DEFAULT_MOLE_WT, "86.18"
 IDS_FILTERSTRING, "Landfill (*.PRM)|*.PRM|All Files (*.*)|*.*|"
}
```

```
SELECTED_AT_DIALOG DIALOG 6, 15, 278, 227
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CLASS "bordlg"
CAPTION "Selected Air Pollutants"
FONT 8, "MS Sans Serif"
{
 DEFPUSHBUTTON "OK", IDC_OK, 55, 204, 50, 14
 PUSHBUTTON "Cancel", IDC_CANCEL, 115, 204, 50, 14
 PUSHBUTTON "Help", IDC_HELP, 175, 204, 50, 14
 CONTROL "", -1, "BorShade", BSS_GROUP | BSS_CAPTION | BSS_LEFT | WS_CHILD | WS_
VISIBLE, 12, 14, 254, 119
 LTEXT "Name", -1, 22, 49, 25, 8
 EDITTEXT IDC_CHEMNAME, 53, 46, 199, 12
 LTEXT "Molecular Weight", -1, 22, 71, 64, 8
 EDITTEXT IDC_MOLEWT, 92, 68, 52, 12
 LTEXT "Concentration (ppmV)", -1, 22, 101, 77, 8
 EDITTEXT IDC_AT_CODIS, 158, 90, 93, 12
 LTEXT "Codisposal", -1, 103, 92, 38, 8
 LTEXT "No Codisposal", -1, 103, 111, 51, 8
 EDITTEXT IDC_AT_NOCO, 158, 110, 92, 12
 PUSHBUTTON "|<", IDC_STARTBTN, 55, 151, 18, 17
 PUSHBUTTON "<", IDC_REVERSEBTN, 73, 151, 18, 17
 PUSHBUTTON ">", IDC_FORWARDBTN, 91, 151, 18, 17
 PUSHBUTTON ">|", IDC_ENDBTN, 109, 151, 18, 17
 CONTROL "", -1, "BorShade", BSS_HDIP | BSS_LEFT | WS_CHILD | WS_VISIBLE, -2, 18
9, 278, 3
 CONTROL "", -1, "BorShade", BSS_GROUP | BSS_CAPTION | BSS_LEFT | WS_CHILD | WS_
VISIBLE, 12, 146, 254, 28
 PUSHBUTTON "Delete", IDC_DELETEBTN, 135, 151, 42, 17
 PUSHBUTTON "Append", IDC_APPENDBTN, 181, 151, 42, 17
 CONTROL "Entry", -1, "BorStatic", SS_LEFT | WS_CHILD | WS_VISIBLE | WS_GROUP, 8
6, 21, 21, 8
 CONTROL "", -1, "BorShade", BSS_GROUP | BSS_CAPTION | BSS_LEFT | WS_CHILD | WS_
VISIBLE, 111, 18, 30, 14
 CONTROL "of", -1, "BorStatic", SS_LEFT | WS_CHILD | WS_VISIBLE | WS_GROUP, 146,
 21, 9, 8
 CONTROL "", -1, "BorShade", BSS_GROUP | BSS_CAPTION | BSS_LEFT | WS_CHILD | WS_
VISIBLE, 161, 18, 30, 14
 LTEXT "", IDC_RECORDNO, 115, 20, 23, 8
 LTEXT "", IDC_RECORDCOUNT, 165, 20, 23, 8
}



STRINGTABLE
{
 AIRTOXICSRC, "1,1,1-Trichloroethane;133.41;0.48;0.48"
 21, "1,1,2,2-Tetrachloroethane;167.85;1.11;1.11"
 22, "1,1,2-Trichloroethane;133.41;0.10;0.10"
 23, "1,1-Dichloroethane;98.96;2.35;2.35"
 24, "1,1-Dichloroethene;96.94;0.201;0.201"
 25, "1,2-Dichloroethane;98.96;0.407;0.407"
 26, "1,2-Dichloropropane;112.99;0.171;0.171"
 27, "2-Propanol;60.01;50.06;50.06"
 28, "Acetone;58.08;7.01;7.01"
 29, "Acrylonitrile;53.06;11.5;11.5"
 30, "Benzene;78.12;1.91;30.0"
 31, "Bromodichloromethane;163.83;3.13;3.13"
 32, "Butane;58.12;5.03;5.03"
 33, "Carbon Disulfide;76.14;0.583;0.583"
```



```
 34, "Carbon Monoxide;28.01;309.32;309.32"
 35, "Carbon Tetrachloride;153.84;0.04;0.04"
 36, "Carbonyl Sulfide;60.07;0.49;0.49"
 37, "Chlorobenzene;112.56;0.254;0.254"
 38, "Chlorodifluoromethane;86.47;1.21;1.21"
 39, "Chloroethane;64.52;1.37;1.37"
 40, "Chloroform;119.38;0.019;0.019"
 41, "Chloromethane;50.49;1.21;1.21"
 42, "Dichlorobenzene;147.00;0.213;0.213"
 43, "Dichlorodifluoromethane;120.91;15.7;15.7"
 44, "Dichlorofluoromethane;102.92;4.4;4.4"
 45, "Dichloromethane;84.93;14.30;14.30"
 46, "Dimethyl Sulfide;62.13;7.82;7.82"
 47, "Ethane;30.07;1105;1105"
 48, "Ethanol;46.08;27.2;27.2"
 49, "Ethylbenzene;106.17;4.61;4.61"
 50, "Ethyl Mercaptan;62.13;2.28;2.28"
 51, "Fluorotrichloromethane;137.37;0.756;0.756"
 52, "Hexane;86.18;6.57;6.57"
 53, "Hydrogen Sulfide;34.08;35.5;35.5"
 54, "Mercury;200.61;0.107;0.107"
 55, "Methyl Ethyl Ketone;72.11;7.09;7.09"
 56, "Methyl Isobutyl Ketone;100.16;1.21;1.21"
 57, "Methyl Mercaptan;48.11;4.33;4.33"
 58, "Pentane;72.15;3.29;3.29"
 59, "Perchloroethylene;165.83;3.73;3.73"
 60, "Propane;44.10;11.1;11.1"
 61, "Toluene;92.14;39.3;165.0"
 62, "Trichloroethene;131.29;2.82;2.82"
 63, "t-1,2-Dichloroethene;96.94;2.84;2.84"
 64, "Vinyl Chloride;62.50;7.34;7.34"
 65, "Xylene;106.17;12.1;12.1"
 66, "END;0;0;0"
}


REPORT_DIALOG DIALOG 6, 15, 192, 166
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CLASS "bordlg"
CAPTION "Select an Emitted Substance"
FONT 8, "MS Sans Serif"
{
 DEFPUSHBUTTON "OK", IDC_OK, 11, 142, 50, 14
 PUSHBUTTON "Cancel", IDC_CANCEL, 71, 142, 50, 14
 PUSHBUTTON "Help", IDC_HELP, 131, 142, 50, 14
 LISTBOX IDC_CHEMLIST, 11, 12, 169, 111, LBS_NOTIFY | WS_BORDER | WS_BORDER | WS
_VSCROLL
 CONTROL "", -1, "BorShade", BSS_HDIP | BSS_LEFT | WS_CHILD | WS_VISIBLE, 0, 131
, 192, 2
}



LANDGRAPH_ICON ICON "landgph.ico"




LANDTEXT_ICON ICON "landtext.ico"

ESTIMATOR_DIALOG DIALOG 6, 15, 202, 131
STYLE WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CLASS "bordlg"
CAPTION "Refuse Estimator"
FONT 8, "MS Sans Serif"
```

```
{
 DEFPUSHBUTTON "Estimate", IDC_OK, 16, 109, 50, 14
 PUSHBUTTON "Ok", IDC_CANCEL, 76, 109, 50, 14
 PUSHBUTTON "Help", IDC_HELP, 136, 109, 50, 14
 CONTROL "", -1, "BorShade", BSS_HDIP | BSS_LEFT | WS_CHILD | WS_VISIBLE, 0, 100
, 208, 2
 CONTROL "", -1, "BorShade", BSS_GROUP | BSS_CAPTION | BSS_LEFT | WS_CHILD | WS_
VISIBLE, 10, 11, 182, 79
 LTEXT "Landfill Size (Acres)", -1, 21, 19, 71, 8
 EDITTEXT IDC_LANDSIZE, 103, 18, 76, 12
 LTEXT "Landfill Depth (Feet)", -1, 21, 41, 71, 8
 EDITTEXT IDC_LANDDEPTH, 103, 39, 76, 12
 LTEXT "Estimated Refuse In Place (Mg)", -1, 21, 65, 77, 19
 CONTROL "", -1, "BorShade", BSS_GROUP | BSS_CAPTION | BSS_LEFT | WS_CHILD | WS_
VISIBLE, 103, 65, 76, 16
 LTEXT "", IDC_ESTIMATE, 107, 69, 68, 8
}

ACCEL ACCELERATORS
{
 VK_F4, LAND_MENU_EXIT, VIRTKEY, ALT
 VK_INSERT, LAND_MENU_COPY, VIRTKEY, CONTROL
 VK_DELETE, LAND_MENU_CUT, VIRTKEY, SHIFT
 VK_INSERT, LAND_MENU_PASTE, VIRTKEY, SHIFT
}


CLOSURE_DIALOG DIALOG 6, 15, 194, 123
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CLASS "bordlg"
CAPTION "Closure Year"
FONT 8, "MS Sans Serif"
{
 DEFPUSHBUTTON "OK", IDC_OK, 12, 101, 50, 14
 PUSHBUTTON "Cancel", IDC_CANCEL, 72, 101, 50, 14
 PUSHBUTTON "Help", IDC_HELP, 132, 101, 50, 14
 CONTROL "", -1, "BorShade", BSS_HDIP | BSS_LEFT | WS_CHILD | WS_VISIBLE, 0, 89,
 194, 2
 CONTROL "", -1, "BorShade", BSS_GROUP | BSS_CAPTION | BSS_LEFT | WS_CHILD | WS_
VISIBLE, 11, 12, 171, 63
 CONTROL "System Calculated", IDC_RADIOCALC, "BorRadio", BS_AUTORADIOBUTTON | WS
_CHILD | WS_VISIBLE | WS_TABSTOP, 54, 29, 75, 10
 CONTROL "User Specified:", IDC_RADIOUSER, "BorRadio", BS_AUTORADIOBUTTON | WS_C
HILD | WS_VISIBLE | WS_TABSTOP, 54, 49, 67, 10
 EDITTEXT IDC_VALUEEDIT, 121, 47, 22, 13
}

CURRENT_PARAMS_DIALOG DIALOG 8, 55, 268, 83
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CLASS "bordlg"
CAPTION "Active Model Parameters"
FONT 8, "MS Sans Serif"
{
 DEFPUSHBUTTON "OK", IDC_OK, 109, 64, 50, 14
 CONTROL "", -1, "BorShade", BSS_GROUP | BSS_CAPTION | BSS_LEFT | WS_CHILD | WS_
VISIBLE, 8, 8, 250, 42
 CONTROL "", -1, "BorShade", BSS_HDIP | BSS_LEFT | WS_CHILD | WS_VISIBLE, 0, 58,
 268, 2
 LTEXT "Methane Generation Rate (1/yr):", -1, 14, 15, 157, 8
 LTEXT "Methane Generation Potential (m^3/Mg):", -1, 14, 34, 151, 9
 CONTROL "", -1, "BorShade", BSS_GROUP | BSS_CAPTION | BSS_LEFT | WS_CHILD | WS_
VISIBLE, 190, 13, 63, 12
 CONTROL "", -1, "BorShade", BSS_GROUP | BSS_CAPTION | BSS_LEFT | WS_CHILD | WS_
VISIBLE, 190, 32, 63, 12
 LTEXT "", IDC_KVALUE, 195, 15, 56, 8
 LTEXT "", IDC_LVALUE, 195, 34, 56, 8
```



```
}

AP42WARN_DIALOG DIALOG 6, 15, 194, 119
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CLASS "bordlg"
CAPTION "AP-42 Alert"
FONT 8, "MS Sans Serif"
{
 DEFPUSHBUTTON "OK", IDC_OK, 72, 96, 50, 14
 CONTROL "", -1, "BorShade", BSS_HDIP | BSS_LEFT | WS_CHILD | WS_VISIBLE, 0, 85,
 194, 2
 CONTROL "", -1, "BorShade", BSS_GROUP | BSS_CAPTION | BSS_LEFT | WS_CHILD | WS_
VISIBLE, 9, 9, 174, 64
 CTEXT "The AP-42 default parameters were current as of December 1994.  However,
 these values are currently under revision and will continue to be updated.  The
refore, you are cautioned to insure that the values used in determining emission
s for recovery purposes reflect the most current and reasonable values.", -1, 14
, 13, 164, 56
}


PRTABORT_DIALOG DIALOG 6, 15, 126, 62
STYLE WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CLASS "bordlg"
CAPTION "Printing"
FONT 8, "MS Sans Serif"
{
 DEFPUSHBUTTON "Cancel", IDC_CANCEL, 38, 37, 50, 14
 CONTROL "", -1, "BorShade", BSS_GROUP | BSS_CAPTION | BSS_LEFT | WS_CHILD | WS_
VISIBLE, 11, 7, 104, 21
 CTEXT "Press Cancel to Abort", -1, 17, 13, 92, 8
}
```

Appendix E

Windows Version Source Code Listings

```
// ============================================================================
//                                DATADEF.H
// ============================================================================
// ============================================================================
//                         Data structures definitions
// ============================================================================
// ============================================================================
//                             Revision History
// ============================================================================
//  09/30/94    Radian(RLB)              Initial Revision
// ============================================================================

#ifndef __DATADEF__
#define __DATADEF__

typedef struct {
               double l0, k;
               double PercentCH4, PercentCO2;
               double NMOC, MoleWt;
               double Temp, Pressure;
               } MODELPARAMETERS;

typedef struct {
               char Name[50];
               double MoleWt;
               double EmFact[2];
               } AIRTOXIC;

typedef struct {
               int AirToxicCount;
               HGLOBAL hAirToxic;     /* Memory Handle to AIRTOXIC array */
               } AIRTOXICLIST;

typedef struct {
               int PeriodCount;
               HGLOBAL hPeriod;      /* Memory Handle to Period List (doubles) */
               } PERIODLIST;

typedef struct {
               int YearOpened, YearCurrent, YearClosed;
               double Capacity;
               double AcceptRate, LastAcceptRate;
               PERIODLIST OpRecord;
               } OPPARAMETERS;

typedef struct {
               double WellDepth, ROI, Density;
               double Fraction, FlowRate, RefuseAge;
               } METHOD2E;


typedef struct {
               int DefaultsMode;
               int LandfillType;
               int MaxYears;
               double Userk;
               double UserL0;
               double UserNMOC;
               } MODELLIMITS;


typedef struct {
               int L0Choice;
               int kChoice;
               int NMOCChoice;
               int LandfillType;
```



```
               int DefaultsMode;
               } PARAMETERINDEX;


#define REGULATORY       0x00
#define AP42             0x01
#define USER             0x02
#define CALCULATED       0x03

#define NOCODISPOSAL     0x00
#define CODISPOSAL       0x10
#define UNKNOWN          0x20

extern MODELLIMITS Limits;
extern MODELPARAMETERS Model;
extern METHOD2E Method2E;

#endif
```

```
// ========================================================================
//                                GRAPHICS.H
// ========================================================================
// ========================================================================
//                    Graphics output data structures
// ========================================================================
// ========================================================================
//                            Revision History
// ========================================================================
//  09/30/94    Radian(RLB)              Initial Revision
// ========================================================================
typedef struct {
              char fontName[80];
              char primaryTitle[80], secondaryTitle[80];
              } TITLESTRUCT;

typedef struct {
               double axisStart, axisEnd;
               double min, max;
               double minorInterval;
               int majorInterval;
               char axisTitle[80];
               char labelFormat[30];
               } AXISSTRUCT;

typedef struct {
               double x, y;
               } POINTSTRUCT;

typedef struct {
               int style;
               int lineStyle;
               COLORREF lineColor, pointColor;
               int pointCount;
               HGLOBAL hPoints;
               } SERIESSTRUCT;


typedef struct {
               TITLESTRUCT titles;
               AXISSTRUCT xAxis, yAxis;
               int seriesCount;
               HGLOBAL hSeries;
               } GRAPHSTRUCT;


int WriteGraphTitle(HDC hdc, RECT *rect, HGLOBAL hPtr);
int WriteAxisTitle(HDC hdc, RECT *rect, HGLOBAL hPtr);
int CalculateAxisDim(HDC hdc, RECT *rect, HGLOBAL hPtr);
int DrawAxesLines(HDC hdc, RECT *rect, HGLOBAL hPtr);
int DrawXAxisTicks(HDC hdc, RECT *rect, HGLOBAL hPtr);
int DrawYAxisTicks(HDC hdc, RECT *rect, HGLOBAL hPtr);
int DrawGraphSeries(HDC hdc, RECT *rect, HGLOBAL hPtr);

long FAR PASCAL WndProc(HWND hWnd, WORD iMessage, WPARAM wParam, LPARAM lParam);
BOOL CALLBACK AboutDlgProc(HWND hWndDlg, UINT msg, WPARAM wParam,
                           LPARAM lParam);
BOOL CALLBACK PrintDlgProc(HWND hWndDlg, UINT msg, WPARAM wParam,

 LPARAM lParam);

HWND  CreateGraphWindow(HINSTANCE hInstance, char *Caption);
void  SetGraphFont(HWND hWnd, char *fontName);
void  SetGraphTitles(HWND hWnd, char *PrimaryTitle,
                                          char *SecondaryTitle);
void  SetGraphSeries(HWND hWnd, HGLOBAL hSeries, int count);
```



```
void  SetAxisTitle(HWND hWnd, int axis, char *title);
void  SetAxisLabelFormat(HWND hWnd, int axis, char *format);
void  SetAxisScale(HWND hWnd, int axis, double min, double max);
void  SetAxisTicks(HWND hWnd, int axis, double minorInterval,
                                                          int ma

jorInterval);



#define X_AXIS 0
#define Y_AXIS 1
```

```
// ========================================================================
//                                HELP[illegible].C
// ========================================================================
// ========================================================================
//                        Constants for context help
// ========================================================================
// ========================================================================
//                            Revision History
// ========================================================================
//  09/30/94    Radian(RLB)              Initial Revision
// ========================================================================

#define  ParametersUser 0x0001
#define  ParametersCalc 0x0002
#define  ParametersAirToxics  0x0003
#define  ATEditor       0x0004
#define  ParametersClosure 0x0005
#define  FileOptions    0x0006
#define  ReportSel      0x0007
```

```
// ============================================================================
//                                LANDOPER.H
// ============================================================================
// ============================================================================
//       Data structure for a landfill study tagged to oper windows
// ============================================================================
// ============================================================================
//                            Revision History
// ============================================================================
//  09/30/94     Radian(RLB)              Initial Revision
// ============================================================================

#ifndef __LANDOPER_H_
#define __LANDOPER_H_

/* Data Structure for the data assigned to an MDI window */
/* This data holds copies of all the stuff necessary to model */
/* a landfill */

typedef struct  {
                char Filename[MAXPATH];
                MODELPARAMETERS Model;
                METHOD2E Method2E;
                OPPARAMETERS Parameters;
                PARAMETERINDEX Choices;
                PERIODLIST Periods;
                AIRTOXICLIST AirToxics;
                int CellIndex, CellOffset;
                int CellCount;
                int Mark;
                } OPERDATASTRUCT;

#endif
```

```
// ==========================================================================
//                                LANDRES.H
// ==========================================================================
// ==========================================================================
//                           Resource constants
// ==========================================================================
// ==========================================================================
//                           Revision History
// ==========================================================================
//  09/30/94    Radian(RLB)              Initial Revision
// ==========================================================================

#define __LANDRES_H_

/* File Selection Options */
#define LAND_MENU_NEW           110
#define LAND_MENU_OPEN          120
#define LAND_MENU_SAVE          130
#define LAND_MENU_SAVEAS            140
#define LAND_MENU_PRINT         150
#define LAND_MENU_EXIT          160

/* Edit Selection Options */
#define LAND_MENU_CUT           210
#define LAND_MENU_COPY          220
#define LAND_MENU_PASTE         230

/* Defaults Selection Options */
#define LAND_MENU_REGULATORY      310
#define LAND_MENU_AP42          320
#define LAND_MENU_SPECIFIC          330
#define LAND_MENU_OPTIONS           340
#define LAND_MENU_CURRENT           350

/* Parameters Selection Options */
#define LAND_MENU_K             410
#define LAND_MENU_DEFAULT_REG_K 411
#define LAND_MENU_DEFAULT_AP42_K      412
#define LAND_MENU_CALC_K            413
#define LAND_MENU_USER_K            414
#define LAND_MENU_L             420
#define LAND_MENU_USER_L            421
#define LAND_MENU_DEFAULT_REG_L 422
#define LAND_MENU_DEFAULT_AP42_L      423
#define LAND_MENU_CODIS         431
#define LAND_MENU_NOCO          432
#define LAND_MENU_AIRTOXICS     440
#define LAND_MENU_CLOSURE       450

/* Reports Selection Options */
#define LAND_MENU_TEXT          510
#define LAND_MENU_GRAPHICS          520

/* Utilties Selection Options */
#define LAND_MENU_ESTIMATE          610
#define LAND_MENU_CONVERT           620

/* Window Selection Options */
#define LAND_MENU_TILE_VERTICAL 701
#define LAND_MENU_TILE_HORIZONTAL     702
#define LAND_MENU_CASCADE       703
#define LAND_MENU_ICONARRANGE   704

/* Help Selection Options */
#define LAND_MENU_HELP_INDEX      901
#define LAND_MENU_ABOUT         999
```



```
/* Dialog Box Controls */

#define IDC_OK                  101
#define IDC_CANCEL                102
#define IDC_HELP                    998
#define IDC_CALCULATE           103
#define IDC_ATBUTTON            104
#define IDC_STARTBTN            105
#define IDC_ENDBTN              106
#define IDC_REVERSEBTN          107
#define IDC_FORWARDBTN          108
#define IDC_DELETEBTN           109
#define IDC_APPENDBTN           110
#define IDC_PRINT               111
#define IDC_SETUP               112


#define IDC_VALUEEDIT           201
#define IDC_WELLDEPTH           202
#define IDC_ROI                 203
#define IDC_DENSITY                 204
#define IDC_FRACTION                205
#define IDC_LO                      206
#define IDC_FLOWRATE            207
#define IDC_AGE                 208
#define IDC_CH4PERCENT          209
#define IDC_NMOCCONC            210
#define IDC_OPENYEAR            211
#define IDC_CURRENTYEAR         212
#define IDC_CAPACITY            213
#define IDC_CHEMNAME            214
#define IDC_MOLEWT              215
#define IDC_AT_CODIS            216
#define IDC_AT_NOCO             217
#define IDC_LANDSIZE            218
#define IDC_LANDDEPTH           219

#define IDC_RADIOBUTTON1            301
#define IDC_RADIOBUTTON2            302
#define IDC_RADIOBUTTON3            303
#define IDC_RADIOAP42           305
#define IDC_RADIOREG              306
#define IDC_RADIOCALC           306
#define IDC_RADIOUSER           307

#define IDC_VALUETEXT           401
#define IDC_CO2PERCENT          402
#define IDC_RECORDNO              403
#define IDC_RECORDCOUNT         404
#define IDC_ESTIMATE            405
#define IDC_KVALUE              406
#define IDC_LVALUE              407

#define IDC_CHEMLIST            501
#define IDC_LISTBOX1            802

#define SYSMENU_SAVE   600
#define SYSMENU_PRINT  602
#define SYSMENU_ABOUT  603
#define SYSMENU_COPY   604

/* String Table Definitions for Defaults */
#define DEFAULT_CH4     1
#define DEFAULT_CO2     2
#define DEFAULT_REGULATORY_LO   3
```

```
#define DEFAULT_REGULATORY_K    4
#define DEFAULT_REGULATORY_NMOC 5
#define DEFAULT_AP42_LO 6
#define DEFAULT_AP42_K  7
#define DEFAULT_AP42_NMOC_CODIS 8
#define DEFAULT_AP42_NMOC_NOCO  9
#define DEFAULT_WELL_DEPTH      10
#define DEFAULT_INFLUENCE_RADIUS        11
#define DEFAULT_REFUSE_DENSITY  12
#define DEFAULT_DECOMP_FRACTION 13
#define DEFAULT_FLOW_RATE       14
#define DEFAULT_REFUSE_AGE      15
#define DEFAULT_TEMP    16
#define DEFAULT_PRESSURE        17
#define DEFAULT_MOLE_WT 18
#define IDS_FILTERSTRING  19
#define AIRTOXICSRC  20
#define IDS_WMFFILTER     70
#define IDS_TEXTFILTER    71

/* Operating Defines */

#define DEFAULT_K           0x02
#define DEFAULT_LO          0x04

#define USER_MODE           0x08
#define AP42_MODE           0x10
#define REGULATORY_MODE     0x20

/* Defines to identify the various MDI windows */

#define NULLWINDOW          0x0000
#define OPERWINDOW          0x0001
#define GRAPHWINDOW         0x0002
#define TEXTWINDOW          0x0003

#define AM_WINDOWACTIVE     WM_USER + 101
#define AM_GETACTIVETOOL    WM_USER + 102
#define AM_MDISAVESTATE     WM_USER + 103
#define AM_MDIRESTORESTATE  WM_USER + 104
#define AM_GETCURRENTENTRY  WM_USER + 105
#define AM_ENTRYSELECTED    WM_USER + 106
#define AM_PERIODCHANGED    WM_USER + 107
#define AM_CONFIRMDATA      WM_USER + 108
#define AM_VALUESCHANGED    WM_USER + 109
#define AM_GETOPERHANDLE    WM_USER + 110
#define AM_UPDATEFIELDS     WM_USER + 111
#define AM_FLUSHFIELDS      WM_USER + 112
#define AM_SAVE             WM_USER + 113
#define AM_PRINT            WM_USER + 114
#define AM_COPY             WM_USER + 115
#define AM_CUT              WM_USER + 116
#define AM_PASTE            WM_USER + 117

#define CNT_SIZEX  0x0100L
#define CNT_SIZEY  0x0200L
```

```
// ========================================================================
//                              LANDWIN.H
// ========================================================================
// ========================================================================
//                  Main include with function prototypes
// ========================================================================
// ========================================================================
//                            Revision History
// ========================================================================
//  09/30/94    Radian(RLB)             Initial Revision
// ========================================================================

#define __LANDWIN_H_

#ifndef __WINDOWS_H
#include <windows.h>
#endif

#ifndef __DATADEF__
#include "datadef.h"
#endif

#ifndef __LANDRES_H_
#include "landres.h"
#endif

#ifndef __DIR_H
#include <dir.h>
#endif

#ifndef __LANDOPER_H_
#include "landoper.h"
#endif

#ifndef __RESULTS_H_
#include "results.h"
#endif


/* Macro Definitions */

#define MDIGETACTIVE() \
        SendMessage(RetrieveClientWindow(), WM_MDIGETACTIVE, 0, 0)
#define MDIGETOPERDATA(x) \
        (HGLOBAL) SendMessage(x, AM_GETOPERHANDLE, 0, 0)


/* Global Data Structure Declaration */

extern char StudyName[MAXPATH];
extern MODELLIMITS Limits;


/* Prototypes */
void OperatingParameters(void);
BOOL GetOldStudy(void);
BOOL GetNewStudy(void);
void SaveStudy(void);
void SaveAsStudy(void);


void CenterWindow(HWND hWndCenter);
void ClearMessages(void);
HWND CreateOperClientWindow(char *title,
                        int x, int y, int width, int height,
                        DWORD style);
```



```
HWND CreateGraphClientWindow(char *title,
                        int x, int y, int width, int height,
                        DWORD style);
HWND CreateTextClientWindow(char *title,
                        int x, int y, int width, int height,
                        DWORD style);
HINSTANCE RetrieveAppInstance(void);
HWND RetrieveClientWindow(void);
HWND RetrieveFrameWindow(void);

void LoadModelLimits(void);
void SaveModelLimits(void);
void LoadModelDefaults(HWND hWnd);
void SetModelDefaults(HWND hWnd);

double ResourceDouble(int ResourceInt);

void RegisterLandwinCustom(void);
void UnregisterLandwinCustom(void);

void RegisterOperClass(void);
void UnregisterOperClass(void);

void CustomizeControl(HWND hWndDlg, int ControlID, FARPROC Procedure);
void UnCustomizeControl(HWND hWndDlg);
void CustomizeWindowControl(HWND hWnd, FARPROC Procedure);
void UnCustomizeWindowControl(HWND hWnd);
LRESULT CALLBACK FPNumberEditProc(HWND hWnd, UINT iMessage, WPARAM wParam, LPARA
M lParam);
LRESULT CALLBACK IntNumberEditProc(HWND hWnd, UINT iMessage, WPARAM wParam, LPAR
AM lParam);

// Prototype for function in VALIDATE.C

BOOL ValidateData(void);

// Prototype for function in CALCMOD.C

HGLOBAL CalculationModule(int ChemIndex);

// Prototype for function in BLDRPT.C

HGLOBAL BuildReport(OPERDATASTRUCT *OperData, RESULTS *Results);

// Prototype for function in OPER.C

void MemAllocError(void);

// Prototype for function for custom controls in SUBCLASS.C

void Draw3DFrame(HWND hWnd, int iStyle);

#define UP_3D    1
#define DOWN_3D  2

// Prototype for function in INDIVAT.C

int GetIndivATInfo(HWND hWnd, HINSTANCE hInst);

// Prototype for function in AIRTOXIC.C

int GetATInfo(HWND hWnd, HINSTANCE hInst);

// Prototype for function in CLOSURE.C

int GetClosureYear(HWND hWnd, HINSTANCE hInst);
```

```
// Prototype for function in RPRTSEL.C

int GetSubstanceSelection(HWND hWnd, HINSTANCE hInst);

// Prototype for function in CALCK.C

int CalculateKValue(HWND hWnd, HINSTANCE hInst);

// Prototype for function in LANDWIN.C

int WhichMDIType(HWND hWnd);

// Function in AP42WARN.C
void DisplayAP42Warn(void);
```

```
// ============================================================================
//                                  RESULTS.H
// ============================================================================
// ============================================================================
//                    Data structure for handling results
// ============================================================================
// ============================================================================
//                              Revision History
// ============================================================================
//  09/30/94     Radian(RLB)              Initial Revision
// ============================================================================
#ifndef __RESULTS_H_
#define __RESULTS_H_

#ifndef __LANDWIN_H_
#include "landwin.h"
#endif

typedef struct {
               int YearStart;
               int YearClosed;
               int YearCount;
               double AcceptanceRate;
               int AirToxicType;
               char Name[50];
               HGLOBAL hRate;
               } RESULTS;

#endif
```

```
// ========================================================================
//                                 ABOUT.C
// ========================================================================
// ========================================================================
//    These are the function prototypes for the functions in str_ops.c
// ========================================================================
// ========================================================================
//                            Revision History
// ========================================================================
//  09/30/94    Radian(RLB)             Initial Revision
// ========================================================================

char *find_noblank(char *);
char *strip_spaces(char *);
char *strip_last_asterisk(char *);
int remove_control(char *);
char *strupper(char *, char *);
char *strlower(char *, char *);
int is_string_number(char *);
int is_string_pos_number(char *);
int is_string_factor(char *);
char *pad_string(char *, int);
int blank_string(char *, int);
```

```
// ========================================================================
//                                TEXTRPT.H
// ========================================================================
// ========================================================================
//             Data structure used on the text report windows
// ========================================================================
// ========================================================================
//                           Revision History
// ========================================================================
//  09/30/94    Radian(RLB)              Initial Revision
// ========================================================================

#define PAGE_WIDTH 80

typedef struct {
               int LineCount;
               HGLOBAL hReportText;
               } REPORTSTRUCT;
```

```
// ========================================================================
//                                 ABOUT.C
// ========================================================================
// ========================================================================
//           Source code for displaying the ABOUT dialog box.
// ========================================================================
// ========================================================================
//                            Revision History
// ========================================================================
//  09/30/94    Radian(RLB)              Initial Revision
// ========================================================================

#include <windows.h>

#include "landwin.h"

BOOL CALLBACK DialogProc(HWND hWndDlg, unsigned uiMsg, WORD wParam,
                         LONG lParam);

void DisplayAbout(HWND hWnd, HINSTANCE hInst)
{
FARPROC DlgProcInstance;
int ret_value;

   DlgProcInstance = MakeProcInstance(DialogProc, hInst);
   DialogBox(hInst, "ABOUT_DIALOG", hWnd, DlgProcInstance);
   FreeProcInstance(DlgProcInstance);
}


#pragma argsused

static BOOL CALLBACK DialogProc(HWND hWndDlg, unsigned uiMsg, WORD wParam,
                                LONG lParam)
{
   switch (uiMsg)
      {
      case WM_INITDIALOG:
         CenterWindow(hWndDlg);
         return TRUE;

      case WM_COMMAND:
         switch (LOWORD(wParam))
            {
            case IDC_OK:
               EndDialog(hWndDlg, TRUE);
               return TRUE;

            default:
               return FALSE;
            }

         default:
            return FALSE;
         }
}
```

```
// ============================================================================
//                                AIRTOXIC.C
// ============================================================================
// ============================================================================
//  Code for displaying and allowing entry on information relating to
//  the pollutant emissions from the landfill.
// ============================================================================
// ============================================================================
//                            Revision History
// ============================================================================
//  09/30/94    Radian(RLB)              Initial Revision
// ============================================================================

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

#include "landwin.h"
#include "str_ops.h"
#include "helpdef.h"


BOOL CALLBACK DialogProc(HWND hWndDlg, unsigned uiMsg, WORD wParam,
                         LONG lParam);


int GetATInfo(HWND hWnd, HINSTANCE hInst)
{
FARPROC DlgProcInstance;
int ret_value;

   DlgProcInstance = MakeProcInstance(DialogProc, hInst);
   ret_value = DialogBox(hInst, "AIR_TOXICS_DIALOG", hWnd, DlgProcInstance);
   FreeProcInstance(DlgProcInstance);

   return ret_value;
}



#pragma argsused

static BOOL CALLBACK DialogProc(HWND hWndDlg, unsigned uiMsg, WORD wParam,
                                LONG lParam)
{
char string[30];
HWND ActiveWindow;
OPERDATASTRUCT *OperData;
double value;
BOOL ReloadFlag;
void SetDialogChoices(HWND hWndDlg);
void SetNMOCValueType(int Type);
void ValidationError(char *string);
void LoadValues(HWND hWndDlg, WORD id);
void SaveValues(HWND hWndDlg);
LRESULT CALLBACK ATFPNumberEditProc(HWND hWnd, UINT iMessage, WPARAM wParam,
                                    LPARAM lParam);

   switch (uiMsg)
      {
      case WM_INITDIALOG:

// Center the dialog box
```



```
         CenterWindow(hWndDlg);

// Subclass the control to allow for beveled edges

         CustomizeControl(hWndDlg, IDC_CH4PERCENT, (FARPROC) ATFPNumberEditProc)
;
         CustomizeControl(hWndDlg, IDC_NMOCCONC,   (FARPROC) ATFPNumberEditProc)
;

// Load the default values.  A value of 0 for the second parameter causes all va
lues
// to be loaded at once.

         LoadValues(hWndDlg, 0);


// Function to set radio buttons depending on the choices in the initialization
// structure.

         SetDialogChoices(hWndDlg);

         return TRUE;

// Message to force field validation.  Sent from a control.  If value fails vali
dation,
// the edit fields are updated to reflect the correct values.

      case AM_CONFIRMDATA:
         switch (wParam)
            {
            case IDC_CH4PERCENT:
               GetDlgItemText(hWndDlg, IDC_CH4PERCENT, string, sizeof(string));
               value = strtod(string, NULL);

               ReloadFlag = FALSE;
               if (value <= 0)
                  {
                  ValidationError("Percentage of methane cannot be less than or
equal to zero");
                  ReloadFlag = TRUE;
                  }

               if (value > 100.0)
                  {
                  ValidationError("Percentage of methane cannot be greater than
100.0");
                  ReloadFlag = TRUE;
                  }

               if (ReloadFlag)
                  {
                  LoadValues(hWndDlg, wParam);
                  GetDlgItemText(hWndDlg, IDC_CH4PERCENT, string, sizeof(string)
);
                  value = strtod(string, NULL);
                  }

               sprintf(string, "%.2f", 100.0 - value);
               SetDlgItemText(hWndDlg, IDC_CO2PERCENT, string);
               return TRUE;

            case IDC_NMOCCONC:
               GetDlgItemText(hWndDlg, IDC_NMOCCONC, string, sizeof(string));
               value = strtod(string, NULL);
```

```
               ReloadFlag = FALSE;
               if (value > 1E6)
                  {
                  ValidationError("Percentage of methane cannot exceed 1,000,000
");
                  ReloadFlag = TRUE;
                  }

               if (ReloadFlag) LoadValues(hWndDlg, wParam);
               return TRUE;
            }


      case WM_COMMAND:

// If the user moves focus from the field, validate the data contained

         if (HIWORD(lParam) == EN_KILLFOCUS)
            {
            PostMessage(hWndDlg, AM_CONFIRMDATA, LOWORD(wParam), 0);
            return FALSE;
            }

         switch (LOWORD(wParam))
            {
            case IDC_OK:
               if (IsDlgButtonChecked(hWndDlg, IDC_RADIOAP42))
                  SetNMOCValueType(AP42);
               else if (IsDlgButtonChecked(hWndDlg, IDC_RADIOREG))
                  SetNMOCValueType(REGULATORY);
               else if (IsDlgButtonChecked(hWndDlg, IDC_RADIOUSER))
                  SetNMOCValueType(USER);

               SaveValues(hWndDlg);

               EndDialog(hWndDlg, TRUE);
               return TRUE;

            case IDC_CANCEL:
               EndDialog(hWndDlg, FALSE);
               return TRUE;

            case IDC_HELP:
               WinHelp(hWndDlg, "LANDWIN.HLP", HELP_CONTEXT, ParametersAirToxics
);
               return TRUE;

            case IDC_RADIOAP42:
            case IDC_RADIOREG:
            case IDC_RADIOUSER:
               if (IsDlgButtonChecked(hWndDlg, IDC_RADIOUSER))
                  EnableWindow(GetDlgItem(hWndDlg, IDC_NMOCCONC), TRUE);
               else EnableWindow(GetDlgItem(hWndDlg, IDC_NMOCCONC), FALSE);
               break;

            case IDC_ATBUTTON:

// Bring up the dialog box to edit the individual pollutant list

               GetIndivATInfo(hWndDlg, RetrieveAppInstance());
               return TRUE;

            default:
               break;
            }
      }
```



```
   return FALSE;
}



void LoadValues(HWND hWndDlg, WORD id)
{
HWND ActiveWindow;
HGLOBAL hOperData = NULL;
OPERDATASTRUCT *OperData;
char string[30];

   ActiveWindow = SendMessage(RetrieveClientWindow(), WM_MDIGETACTIVE,
                              0, 0);
   if (ActiveWindow != NULL)
      hOperData = SendMessage(ActiveWindow, AM_GETOPERHANDLE, 0, 0);

   if (hOperData != NULL)
      OperData = (OPERDATASTRUCT *) GlobalLock(hOperData);

   if (OperData != NULL)
      {
      switch (id)
         {
         default:
         case IDC_CH4PERCENT:
            sprintf(string, "%.2f", OperData->Model.PercentCH4);
            SetDlgItemText(hWndDlg, IDC_CH4PERCENT, (LPCSTR) string);
            if (id == IDC_CH4PERCENT) break;

         case IDC_CO2PERCENT:
            sprintf(string, "%.2f", 100.0 - OperData->Model.PercentCH4);
            SetDlgItemText(hWndDlg, IDC_CO2PERCENT, (LPCSTR) string);
            if (id == IDC_CO2PERCENT) break;

         case IDC_NMOCCONC:
            sprintf(string, "%.2f", OperData->Model.NMOC);
            SetDlgItemText(hWndDlg, IDC_NMOCCONC, (LPCSTR) string);
            if (id == IDC_NMOCCONC) break;
            break;
         }
      }

   if (hOperData != NULL) GlobalUnlock(hOperData);
}



// Sets the values in the data structure memory to the settings on the dialog
// box.


void SaveValues(HWND hWndDlg)
{
HWND ActiveWindow;
HGLOBAL hOperData = NULL;
OPERDATASTRUCT *OperData;
char string[30];

   ActiveWindow = SendMessage(RetrieveClientWindow(), WM_MDIGETACTIVE,
                              0, 0);
   if (ActiveWindow != NULL)
      hOperData = SendMessage(ActiveWindow, AM_GETOPERHANDLE, 0, 0);

   if (hOperData != NULL)
```

```
      OperData = (OPERDATASTRUCT *) GlobalLock(hOperData);

   if (OperData != NULL)
      {
      GetDlgItemText(hWndDlg, IDC_CH4PERCENT, string, sizeof(string));
      OperData->Model.PercentCH4 = strtod(string, NULL);
      OperData->Model.PercentCO2 = 100.0 - OperData->Model.PercentCH4;

      GetDlgItemText(hWndDlg, IDC_NMOCCONC, string, sizeof(string));
      OperData->Model.NMOC = strtod(string, NULL);
      }

   if (hOperData != NULL) GlobalUnlock(hOperData);
}



// Generic error message display function.

static void ValidationError(char *string)
{
   MessageBox(NULL, string, "Validation Error", MB_ICONSTOP | MB_OK);
}



// Changes the displayed radio button states and edit field values based
// on the current state of the oper data area.

void SetDialogChoices(HWND hWndDlg)
{
HWND ActiveWindow;
HGLOBAL hOperData = NULL;
OPERDATASTRUCT *OperData;

   ActiveWindow = SendMessage(RetrieveClientWindow(), WM_MDIGETACTIVE,
                              0, 0);
   if (ActiveWindow != NULL)
      hOperData = SendMessage(ActiveWindow, AM_GETOPERHANDLE, 0, 0);

   if (hOperData != NULL)
      OperData = (OPERDATASTRUCT *) GlobalLock(hOperData);

   if (OperData != NULL)
      {
      EnableWindow(GetDlgItem(hWndDlg, IDC_NMOCCONC), FALSE);

      EnableWindow(GetDlgItem(hWndDlg, IDC_RADIOREG), FALSE);
      EnableWindow(GetDlgItem(hWndDlg, IDC_RADIOAP42), FALSE);
      EnableWindow(GetDlgItem(hWndDlg, IDC_RADIOUSER), FALSE);

      if (OperData->Choices.DefaultsMode == REGULATORY)
         {
         CheckRadioButton(hWndDlg, IDC_RADIOAP42, IDC_RADIOUSER,
                          IDC_RADIOREG);
         EnableWindow(GetDlgItem(hWndDlg, IDC_RADIOREG), TRUE);
         }
      else if (OperData->Choices.DefaultsMode == AP42)
         {
         CheckRadioButton(hWndDlg, IDC_RADIOAP42, IDC_RADIOUSER,
                          IDC_RADIOAP42);
         EnableWindow(GetDlgItem(hWndDlg, IDC_RADIOAP42), TRUE);
         }
      else if (OperData->Choices.DefaultsMode == USER)
```



```
         {
         CheckRadioButton(hWndDlg, IDC_RADIOAP42, IDC_RADIOUSER,
                          IDC_RADIOUSER);

         EnableWindow(GetDlgItem(hWndDlg, IDC_NMOCCONC),  TRUE);

         EnableWindow(GetDlgItem(hWndDlg, IDC_RADIOREG),  TRUE);
         EnableWindow(GetDlgItem(hWndDlg, IDC_RADIOAP42), TRUE);
         EnableWindow(GetDlgItem(hWndDlg, IDC_RADIOUSER), TRUE);
         }
      }

   if (hOperData != NULL) GlobalUnlock(hOperData);
}



// Function locks down the operational data specific data (tagged to a single op
er window)
// and changes the value based on the dialog box settings.


void SetNMOCValueType(int Type)
{
HWND ActiveWindow;
HGLOBAL hOperData = NULL;
OPERDATASTRUCT *OperData;

   ActiveWindow = SendMessage(RetrieveClientWindow(), WM_MDIGETACTIVE,
                              0, 0);
   if (ActiveWindow != NULL)
      hOperData = SendMessage(ActiveWindow, AM_GETOPERHANDLE, 0, 0);

   if (hOperData != NULL)
      OperData = (OPERDATASTRUCT *) GlobalLock(hOperData);

   if (OperData != NULL) OperData->Choices.NMOCChoice = Type;

   if (hOperData != NULL) GlobalUnlock(hOperData);
}



// Function to subclass the control to bevel the edges and to filter keystrokes
// for only those applicable to floating point numbers.

static LRESULT CALLBACK ATFPNumberEditProc(HWND hWnd, UINT iMessage,
                                           WPARAM wParam, LPARAM lParam)
{
HWND ActiveWindow;
char string[30];
LRESULT ret;
void UnCustomizeControl(HWND hWnd);
LRESULT CALLBACK CallSCProc(HWND hWnd, UINT iMessage, WORD wParam,
                            DWORD lParam);
NCCALCSIZE_PARAMS *lpncsp = (NCCALCSIZE_PARAMS *) lParam;
WINDOWPOS *WinPos;

   switch (iMessage)
      {
      case WM_GETDLGCODE:
         ret = CallSCProc(hWnd, iMessage, wParam, lParam);
         if (lParam)
            {
```

```
            if (((MSG *) lParam)->message == WM_KEYDOWN)
               if (((MSG *) lParam)->wParam == VK_RETURN) ret |= DLGC_WANTMESSAG
E;
            }
         return ret;

      case WM_NCDESTROY:
         UnCustomizeControl(hWnd);
         return TRUE;

      case WM_CHAR:
         memset(string, 0, sizeof(string));
         SendMessage(hWnd, WM_GETTEXT, sizeof(string), (long) string);

         *(string + strlen(string)) = (char) wParam;

         if (isprint((char) wParam))
            {
            if (is_string_pos_number(string))
                 return CallSCProc(hWnd, iMessage, wParam, lParam);
            else return TRUE;
            }

         return CallSCProc(hWnd, iMessage, wParam, lParam);

      case WM_NCPAINT:
         Draw3DFrame(hWnd, DOWN_3D);
         return TRUE;

      case WM_NCCALCSIZE:
         if (! (BOOL) wParam) return FALSE;

         WinPos = lpncsp->lppos;

         CopyRect(((RECT *) lpncsp->rgrc) + 2, lpncsp->rgrc);

         lpncsp->rgrc[1].left += 2;
         lpncsp->rgrc[1].right -= 2;
         lpncsp->rgrc[1].top += 2;
         lpncsp->rgrc[1].bottom -= 2;

         lpncsp->rgrc[2].left += 2;
         lpncsp->rgrc[2].right -= 2;
         lpncsp->rgrc[2].top += 2;
         lpncsp->rgrc[2].bottom -= 2;

         CopyRect(lpncsp->rgrc, ((RECT *) lpncsp->rgrc) + 2);

         lpncsp->rgrc[0].left = WinPos->x + 2;
         lpncsp->rgrc[0].top = WinPos->y + 2;
         lpncsp->rgrc[0].right = WinPos->cx - 2;
         lpncsp->rgrc[0].bottom = WinPos->cy - 2;

         return WVR_VALIDRECTS & WVR_HREDRAW & WVR_VREDRAW;

      case WM_PAINT:
         CallSCProc(hWnd, iMessage, wParam, lParam);
         Draw3DFrame(hWnd, DOWN_3D);
         return FALSE;

       case WM_KEYDOWN:
         switch (wParam)
              {
              case VK_RETURN:
                 PostMessage(GetParent(hWnd), AM_CONFIRMDATA,
                                 GetDlgCtrlID(hWnd), 0);
```



```
                 return TRUE;

              default:
                 return CallSCProc(hWnd, iMessage, wParam, lParam);
              }

      default:
         return CallSCProc(hWnd, iMessage, wParam, lParam);
      }
]
```

```
// ==========================================================================
//                                  AP42WARN.C
// ==========================================================================
// ==========================================================================
//            Source code for displaying the ABOUT dialog box.
// ==========================================================================
// ==========================================================================
//                            Revision History
// ==========================================================================
//  09/30/94    Radian(RLB)             Initial Revision
// ==========================================================================

#include <windows.h>

#include "landwin.h"

BOOL CALLBACK DialogProc(HWND hWndDlg, unsigned uiMsg, WORD wParam,
                         LONG lParam);

void DisplayAP42Warn(void)
{
HWND hWnd = RetrieveFrameWindow();
HINSTANCE hInst = RetrieveAppInstance();
FARPROC DlgProcInstance;
int ret_value;
static BOOL WarnFlag = FALSE;

   if (WarnFlag) return;

   DlgProcInstance = MakeProcInstance(DialogProc, hInst);
   DialogBox(hInst, "AP42WARN_DIALOG", hWnd, DlgProcInstance);
   FreeProcInstance(DlgProcInstance);

   WarnFlag = TRUE;
}


#pragma argsused

static BOOL CALLBACK DialogProc(HWND hWndDlg, unsigned uiMsg, WORD wParam,
                                LONG lParam)
{
   switch (uiMsg)
      {
      case WM_INITDIALOG:
         CenterWindow(hWndDlg);
         return TRUE;

      case WM_COMMAND:
         switch (LOWORD(wParam))
            {
            case IDC_OK:
               EndDialog(hWndDlg, TRUE);
               return TRUE;

            default:
               return FALSE;
            }

         default:
            return FALSE;
         }
}
```

```
// ==================================================================
//                              BLDRPT.C
// ==================================================================
// ==================================================================
//            Source code for building the report text.
// ==================================================================
// ==================================================================
//                          Revision History
// ==================================================================
//  09/30/94    Radian(RLB)              Initial Revision
// ==================================================================
#include <stdio.h>
#include <stdarg.h>
#include <string.h>
#include <windows.h>

#include "landwin.h"
#include "results.h"
#include "textrpt.h"
#include "str_ops.h"

/* Function creates a REPORTSTRUCT Global */


HGLOBAL BuildReport(OPERDATASTRUCT *OperData, RESULTS *Results)
{
REPORTSTRUCT *ReportStruct;
AIRTOXIC *AirToxic = (AIRTOXIC *) GlobalLock(OperData->AirToxics.hAirToxic);
double *Refuse, *VolEmissions, *MassEmissions;
double sum;
HGLOBAL hReportStruct;
char *ptr;
char (*string)[PAGE_WIDTH + 5];
int i, j;
int line = 0;
void CenterPrintString(char *Dest, char *String, int Width);
void AtSay(char *String, int loc, char *Text);
void AtSayPrintf(char *String, int loc, char *Format, ...);

   hReportStruct = GlobalAlloc(GHND, sizeof(REPORTSTRUCT));
   if (hReportStruct == NULL) return NULL;

   ReportStruct = (REPORTSTRUCT *) GlobalLock(hReportStruct);

   ReportStruct->LineCount = 28 + Results->YearCount;
   ReportStruct->hReportText = GlobalAlloc(GHND,
                              ReportStruct->LineCount
                              * (PAGE_WIDTH + 5) * sizeof(char));// This many cha
rs
   if (ReportStruct->hReportText == NULL) return NULL;


   string = (char (*)[PAGE_WIDTH + 5]) GlobalLock(ReportStruct->hReportText);

/* Model Parameters Section */

   memset(string[line++], '=', PAGE_WIDTH);
   CenterPrintString(string[line++], "Model Parameters", PAGE_WIDTH);
   memset(string[line++], '=', PAGE_WIDTH);

   sprintf(string[line], "Lo : %.2f m^3 / Mg", OperData->Model.l0);
   if (OperData->Choices.L0Choice == USER)
      strcat(string[line], " ***** User Mode Selection *****");
   line++;

   sprintf(string[line], "k : %.4f 1/yr", OperData->Model.k);
```



```
   if (OperData->Choices.kChoice == USER)
      strcat(string[line], " ***** User Mode Selection *****");
   else if (OperData->Choices.kChoice == CALCULATED)
      strcat(string[line], " ***** Calculated Value *****");
   line++;

   sprintf(string[line], "NMOC : %.2f ppmv", OperData->Model.NMOC);
   if (OperData->Choices.kChoice == USER)
      strcat(string[line], " ***** User Mode Selection *****");
   line++;

   sprintf(string[line++], "Methane : %.4f %% volume", OperData->Model.PercentCH
4);
   sprintf(string[line++], "Carbon Dioxide : %.4f %% volume", OperData->Model.Pe
rcentCO2);

   if (Results->AirToxicType > 2)
      {
      sprintf(string[line++], "Desired Air Toxic Parameters : %s",
              Results->Name);

      if (OperData->Choices.LandfillType == CODISPOSAL) i = 1;
      else i = 0;

      sprintf(string[line++], "Molecular Wt = %7.2f     Concentration = %13.6f p
pmV",
                  AirToxic[Results->AirToxicType - 2].MoleWt,
                  AirToxic[Results->AirToxicType - 2].EmFact[i]);
      }

   line++;

/* Landfill Operating Parameters Section */

   memset(string[line++], '=', PAGE_WIDTH);
   CenterPrintString(string[line++], "Landfill Parameters", PAGE_WIDTH);
   memset(string[line++], '=', PAGE_WIDTH);

   sprintf(string[line++], "Year Opened : %d    Current Year : %d   Year Closed:
 %d",
              OperData->Parameters.YearOpened, OperData->Parameters.YearCurrent
,
              Results->YearClosed);

   sprintf(string[line++], "Capacity : %.0f Mg", OperData->Parameters.Capacity);

   sprintf(string[line++], "Average Acceptance Rate Required from");
   sprintf(string[line++], "          Current Year to Closure Year : %.2f Mg/year
",
              Results->AcceptanceRate);
   line++;

   memset(string[line++], '=', PAGE_WIDTH);

/* Print the model results */

   CenterPrintString(string[line++], "Model Results", PAGE_WIDTH);
   memset(string[line++], '=', PAGE_WIDTH);

/* Double Array holds Refuse Amounts, Mass Emissions, and Vol Emissions */

   Refuse = (double *) GlobalLock(Results->hRate);
   MassEmissions = Refuse + Results->YearCount;
   VolEmissions = Refuse + 2 * Results->YearCount;

   memset(string[line], ' ', PAGE_WIDTH);
```

```
   AtSayPrintf(string[line++], 49 - strlen(Results->Name) / 2, "%s Emission Rate
",
               Results->Name);

   memset(string[line], ' ', PAGE_WIDTH);
   AtSay(string[line],    0, "Year");
   AtSay(string[line],    9, "Refuse In Place (Mg)");
   AtSay(string[line],   41, "(Mg/yr)");
   AtSay(string[line++], 57, "(Cubic m/yr)");
   memset(string[line++], '=', PAGE_WIDTH);

   for (sum = 0.0, i = 0 ; i < Results->YearCount ; i++)
      {
      memset(string[line], ' ', PAGE_WIDTH);

      AtSayPrintf(string[line], 0, "%4d", Results->YearStart + i);

      sum += Refuse[i];
      AtSayPrintf(string[line], 9, "%13.3E", sum);

      AtSayPrintf(string[line], 37, "%12.3E", MassEmissions[i]);
      AtSayPrintf(string[line++], 56, "%12.3E", VolEmissions[i]);
      }

   GlobalUnlock(OperData->AirToxics.hAirToxic);
   GlobalUnlock(ReportStruct->hReportText);
   GlobalUnlock(hReportStruct);

   return hReportStruct;
}


void CenterPrintString(char *Dest, char *String, int Width)
{
   memset(Dest, ' ', Width);
   memcpy(Dest + (Width - strlen(String)) / 2, String, strlen(String));
}


void AtSay(char *String, int loc, char *Text)
{
   memcpy(String + loc, Text, strlen(Text));
}


void AtSayPrintf(char *String, int loc, char *Format, ...)
{
va_list argptr;
char temp[80];

   va_start(argptr, Format);
   vsprintf(temp, Format, argptr);
   va_end(argptr);

   memcpy(String + loc, temp, strlen(temp));
}
```

```
// ============================================================================
//                                  CALCK.C
// ============================================================================
// ============================================================================
//            Source code for displaying the ABOUT dialog box.
// ============================================================================
// ============================================================================
//                            Revision History
// ============================================================================
//  09/30/94    Radian(RLB)             Initial Revision
// ============================================================================

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

#include "landwin.h"
#include "str_ops.h"
#include "helpdef.h"


BOOL CALLBACK DialogProc(HWND hWndDlg, unsigned uiMsg, WORD wParam,
                         LONG lParam);


int CalculateKValue(HWND hWnd, HINSTANCE hInst)
{
FARPROC DlgProcInstance;
int ret_value;

   DlgProcInstance = MakeProcInstance(DialogProc, hInst);
   ret_value = DialogBox(hInst, "CALCULATE_DIALOG", hWnd, DlgProcInstance);
   FreeProcInstance(DlgProcInstance);

   return ret_value;
}


#pragma argsused

static BOOL CALLBACK DialogProc(HWND hWndDlg, unsigned uiMsg, WORD wParam,
                                LONG lParam)
{
HINSTANCE hInst;
HWND ActiveWindow;
HGLOBAL hOperData;
OPERDATASTRUCT *OperData;
double value;
char string[40];
void LoadCalcValues(HWND hWndDlg);
void SubclassEditControls(HWND hWndDlg);
int GetValidateData(HWND hWndDlg);
double CalculateK(METHOD2E *Method2E, MODELPARAMETERS *Model);

   switch (uiMsg)
      {
      case WM_INITDIALOG:
         CenterWindow(hWndDlg);
         LoadCalcValues(hWndDlg);
         SubclassEditControls(hWndDlg);
         return TRUE;

      case WM_COMMAND:
         switch (LOWORD(wParam))
```



```
         {
         case IDC_OK:
            GetDlgItemText(hWndDlg, IDC_VALUETEXT, (LPSTR) string,
                           sizeof(string));

            value = strtod(string, NULL);
            if (value > 0)
               {
               ActiveWindow = MDIGETACTIVE();
               if (ActiveWindow != NULL)
                  hOperData = MDIGETOPERDATA(ActiveWindow);
               if (hOperData != NULL)
                  OperData = (OPERDATASTRUCT *) GlobalLock(hOperData);

               OperData->Model.k = value;

               GlobalUnlock(hOperData);
               }

            EndDialog(hWndDlg, TRUE);
            return TRUE;

         case IDC_CANCEL:
            EndDialog(hWndDlg, FALSE);
            return TRUE;

         case IDC_HELP:
            WinHelp(hWndDlg, "LANDWIN.HLP", HELP_CONTEXT, ParametersCalc);
            return TRUE;

         case IDC_CALCULATE:
            if (GetValidateData(hWndDlg))
               {
               ActiveWindow = MDIGETACTIVE();

               if (ActiveWindow != NULL)
                  hOperData = MDIGETOPERDATA(ActiveWindow);
               if (hOperData != NULL)
                  OperData = (OPERDATASTRUCT *) GlobalLock(hOperData);

               value = CalculateK(&(OperData->Method2E),
                                  &(OperData->Model));

               GlobalUnlock(hOperData);

               if (value < -10000.0)
                  MessageBox(hWndDlg, "Solution does not converge",
                             "Calculation Failure",
                             MB_OK | MB_ICONEXCLAMATION);
               }
            else value = -99999.0;

            if (value < -10000.0) sprintf(string, "No Solution");
            else sprintf(string, "%lf", value);

            SetDlgItemText(hWndDlg, IDC_VALUETEXT, string);
            return TRUE;

         default:
            break;
         }
      }

   return FALSE;
}
```

```
void LoadCalcValues(HWND hWndDlg)
{
char string[30];
HWND ActiveWindow;
HGLOBAL hOperData;
OPERDATASTRUCT *OperData;

   ActiveWindow = MDIGETACTIVE();

   if (ActiveWindow != NULL) hOperData = MDIGETOPERDATA(ActiveWindow);
   if (hOperData != NULL) OperData = (OPERDATASTRUCT *) GlobalLock(hOperData);

   sprintf(string, "%.4g", OperData->Method2E.WellDepth);
   SetDlgItemText(hWndDlg, IDC_WELLDEPTH, (LPCSTR) string);

   sprintf(string, "%.4g", OperData->Method2E.ROI);
   SetDlgItemText(hWndDlg, IDC_ROI, (LPCSTR) string);

   sprintf(string, "%.4g", OperData->Method2E.Density);
   SetDlgItemText(hWndDlg, IDC_DENSITY, (LPCSTR) string);

   sprintf(string, "%.4g", OperData->Method2E.Fraction);
   SetDlgItemText(hWndDlg, IDC_FRACTION, (LPCSTR) string);

   sprintf(string, "%.4g", OperData->Model.l0);
   SetDlgItemText(hWndDlg, IDC_L0, (LPCSTR) string);

   sprintf(string, "%.4g", OperData->Method2E.FlowRate);
   SetDlgItemText(hWndDlg, IDC_FLOWRATE, (LPCSTR) string);

   sprintf(string, "%.4g", OperData->Method2E.RefuseAge);
   SetDlgItemText(hWndDlg, IDC_AGE, (LPCSTR) string);

   SetDlgItemText(hWndDlg, IDC_VALUETEXT, (LPCSTR) "No Solution");

   GlobalUnlock(hOperData);
}



static int GetValidateData(HWND hWndDlg)
{
char string[30];
void ValidationError(HWND parent, char *text);
HWND ActiveWindow;
HGLOBAL hOperData;
OPERDATASTRUCT *OperData;

   ActiveWindow = MDIGETACTIVE();

   if (ActiveWindow != NULL) hOperData = MDIGETOPERDATA(ActiveWindow);
   if (hOperData != NULL) OperData = (OPERDATASTRUCT *) GlobalLock(hOperData);

   GetDlgItemText(hWndDlg, IDC_WELLDEPTH, string, sizeof(string));
   OperData->Method2E.WellDepth = strtod(string, NULL);
   if (OperData->Method2E.WellDepth <= 0.0)
      {
      ValidationError(hWndDlg, "Well Depth must be greater than zero");
      return FALSE;
      }

   GetDlgItemText(hWndDlg, IDC_ROI, string, sizeof(string));
   OperData->Method2E.ROI = strtod(string, NULL);
   if (OperData->Method2E.ROI <= 0.0)
```



```
      {
      ValidationError(hWndDlg, "Radius of influence must be greater than zero");
      return FALSE;
      }

   GetDlgItemText(hWndDlg, IDC_DENSITY, string, sizeof(string));
   OperData->Method2E.Density = strtod(string, NULL);
   if (OperData->Method2E.Density <= 0.0)
      {
      ValidationError(hWndDlg, "Density must be greater than zero");
      return FALSE;
      }

   GetDlgItemText(hWndDlg, IDC_FRACTION, string, sizeof(string));
   OperData->Method2E.Fraction = strtod(string, NULL);
   if (OperData->Method2E.Fraction <= 0.0)
      {
      ValidationError(hWndDlg, "Refuse fraction must be greater than zero");
      return FALSE;
      }

   GetDlgItemText(hWndDlg, IDC_L0, string, sizeof(string));
   OperData->Model.l0 = strtod(string, NULL);
   if (OperData->Model.l0 <= 0.0)
      {
      ValidationError(hWndDlg, "L0 must be greater than zero");
      return FALSE;
      }

   GetDlgItemText(hWndDlg, IDC_FLOWRATE, string, sizeof(string));
   OperData->Method2E.FlowRate = strtod(string, NULL);
   if (OperData->Method2E.FlowRate <= 0.0)
      {
      ValidationError(hWndDlg, "Flow rate must be greater than zero");
      return FALSE;
      }

   GetDlgItemText(hWndDlg, IDC_AGE, string, sizeof(string));
   OperData->Method2E.RefuseAge = strtod(string, NULL);
   if (OperData->Method2E.RefuseAge <= 0.0)
      {
      ValidationError(hWndDlg, "Refuse age must be greater than zero");
      return FALSE;
      }


   GlobalUnlock(hOperData);
   return TRUE;
}



void SubclassEditControls(HWND hWndDlg)
{
   CustomizeControl(hWndDlg, IDC_WELLDEPTH, (FARPROC) FPNumberEditProc);
   CustomizeControl(hWndDlg, IDC_ROI, (FARPROC) FPNumberEditProc);
   CustomizeControl(hWndDlg, IDC_DENSITY, (FARPROC) FPNumberEditProc);
   CustomizeControl(hWndDlg, IDC_L0, (FARPROC) FPNumberEditProc);
   CustomizeControl(hWndDlg, IDC_FLOWRATE, (FARPROC) FPNumberEditProc);
   CustomizeControl(hWndDlg, IDC_FRACTION, (FARPROC) FPNumberEditProc);
   CustomizeControl(hWndDlg, IDC_AGE, (FARPROC) FPNumberEditProc);
}
```

```
static void ValidationError(HWND parent, char *text)
{
   MessageBox(parent, text, "Validation Error", MB_ICONSTOP | MB_OK);
}
```

```
/* ========================================================================== */
/*                                  CALCMOD.C                                 */
/* ========================================================================== */
/*  This file contains the routines used to calculate the landfill gas        */
/*  emissions.                                                                */
/* ========================================================================== */
/*                             Created under Contract                         */
/*                                       to                                   */
/*                                    EPA/CTC                                 */
/*                                                                            */
/*                                       by                                   */
/*                                                                            */
/*                                  Radian Corporation                        */
/*                              Research Triangle Park, NC                    */
/* ========================================================================== */
/* ========================================================================== */
/*                                Revision History                            */
/* ========================================================================== */
/*    06/03/94   rlb       Initial Implementation                             */
/* ========================================================================== */

#include <stdio.h>
#include <string.h>
#include <math.h>
#include <windows.h>

#include "landwin.h"
#include "results.h"


/* ========================================================================== */
/*               R - Universal Gas Constant  (J / (mole K))                   */
/* ========================================================================== */

#define R   8.3144

/* ========================================================================== */
/*                 This is the main calculation function.                     */
/*  This function takes care of all of the results memory management as       */
/*  well as coordinating the calculation of the emission information.         */
/* ========================================================================== */

void CalcResultsParams(RESULTS *Results, OPERDATASTRUCT *OperData);
void FillRefuseList(OPERDATASTRUCT *OperData, RESULTS *Results, double *Rate);
double mass_to_volume(double mass, double mole_wt, double temp, double pressure)
;
double volume_to_mass(double mass, double mole_wt, double temp, double pressure)
;
double model_function(double k, double l0, double *refuse, int i);


HGLOBAL CalculationModule(int ChemIndex)
{
HWND ActiveWindow;
HGLOBAL hResults, hOperData = NULL;
OPERDATASTRUCT *OperData;
AIRTOXIC *AirToxic;
RESULTS *Results;
int i, index;
double *Rate, *MassEmissions, *VolEmissions;
double ch4_rate, factor;

double model_function();
double mass_to_volume();
double volume_to_mass();
```



```
   ActiveWindow = SendMessage(RetrieveClientWindow(), WM_MDIGETACTIVE,
                                        0, 0);
   if (ActiveWindow != NULL)
      hOperData = SendMessage(ActiveWindow, AM_GETOPERHANDLE, 0, 0);

   if (hOperData != NULL)
      OperData = (OPERDATASTRUCT *) GlobalLock(hOperData);

   if (OperData == NULL) return NULL;

   hResults = GlobalAlloc(GHND, sizeof(RESULTS));
   if (hResults == NULL)
      {
      GlobalUnlock(hOperData);
      return NULL;
      }

   Results = (RESULTS *) GlobalLock(hResults);

   Results->YearStart = OperData->Parameters.YearOpened + 1;

   CalcResultsParams(Results, OperData);

   Results->YearCount = Results->YearClosed - Results->YearStart
                        + Limits.MaxYears;

   AirToxic = (AIRTOXIC *) GlobalLock(OperData->AirToxics.hAirToxic);


   if (ChemIndex == 0) strcpy(Results->Name, "Methane");
   else if (ChemIndex == 1) strcpy(Results->Name, "Carbon Dioxide");
   else if (ChemIndex == 2) strcpy(Results->Name, "NMOC");
   else strcpy(Results->Name, AirToxic[ChemIndex - 3].Name);

   Results->AirToxicType = ChemIndex;

   Results->hRate
         = GlobalAlloc(GHND, 3 * Results->YearCount * sizeof(double));

   Rate = (double *) GlobalLock(Results->hRate);
   MassEmissions = Rate + Results->YearCount;
   VolEmissions = Rate + 2 * Results->YearCount;

   FillRefuseList(OperData, Results, Rate);

/* ========================================================================== */
/*                                 Main Loop                                  */
/*  This loop calculates the methane, carbon dioxide, nmoc, and air toxic     */
/*  emissions (if any) from the year after landfill start until the last      */
/*  desired calculation year (n years past closure).                          */
/* ========================================================================== */

   for (i = 0 ; i < Results->YearCount ; i++)
      {
/* ========================================================================== */
/*              Get the current methane generation rate (Cft/yr)              */
/* ========================================================================== */
      ch4_rate = model_function(OperData->Model.k, OperData->Model.l0,
                                Rate, i);

      if (ChemIndex == 0)
         {
         VolEmissions[i] = ch4_rate;
         MassEmissions[i] = volume_to_mass(VolEmissions[i], 16.04,
                                           OperData->Model.Temp,
                                           OperData->Model.Pressure) / 1000;
```

```
        }
      else if (ChemIndex == 1)
         {
         VolEmissions[i] = ch4_rate
               * OperData->Model.PercentCH4 / OperData->Model.PercentCO2;
         MassEmissions[i] = volume_to_mass(VolEmissions[i], 44.01,
                                           OperData->Model.Temp,
                                           OperData->Model.Pressure) / 1000;
         }
      else if (ChemIndex == 2)
         {
         VolEmissions[i] = ch4_rate * 100.0 * OperData->Model.NMOC
                                    / (1.0E6 * OperData->Model.PercentCH4);
         MassEmissions[i] = volume_to_mass(VolEmissions[i],
                                           OperData->Model.MoleWt,
                                           OperData->Model.Temp,
                                           OperData->Model.Pressure) / 1000;
         }
      else
         {
         if (OperData->Choices.LandfillType == CODISPOSAL)
            factor = AirToxic[ChemIndex - 3].EmFact[0];
         else
            factor = AirToxic[ChemIndex - 3].EmFact[1];

         VolEmissions[i] = ch4_rate * 100.0 * factor
                                    / (1.0E6 * OperData->Model.PercentCH4);
         MassEmissions[i] = volume_to_mass(VolEmissions[i],
                                           AirToxic[ChemIndex - 3].MoleWt,
                                           OperData->Model.Temp,
                                           OperData->Model.Pressure) / 1000;
         }
      }

/* ========================================================================== */
/*                                End Loop                                    */
/* ========================================================================== */

   GlobalUnlock(Results->hRate);
   GlobalUnlock(OperData->AirToxics.hAirToxic);
   GlobalUnlock(hOperData);
   GlobalUnlock(hResults);

   return hResults;
}



void CalcResultsParams(RESULTS *Results, OPERDATASTRUCT *OperData)
{
double AcceptanceRate;
double *Period;
int i, Index;
int MoreYears;

   Period = (double *) GlobalLock(OperData->Periods.hPeriod);

   if (OperData->Parameters.YearClosed == 0)
      {
/* Base the acceptance rate on the average for the period */

//       for (i = 1 ; i < OperData->Periods.PeriodCount ; i++)
//          Sum += (Period[i] - Period[i - 1]);

//       Results->AcceptanceRate = Sum / OperData->Periods.PeriodCount;
```



```
      Results->AcceptanceRate = Period[OperData->Periods.PeriodCount - 1];

      MoreYears = ceil((OperData->Parameters.Capacity
                     - Period[OperData->Periods.PeriodCount - 1])
                        / Results->AcceptanceRate);

      if (MoreYears > Limits.MaxYears) MoreYears = Limits.MaxYears;
      }
   else
      {
      MoreYears = OperData->Parameters.YearClosed
                     - OperData->Parameters.YearCurrent;

      Results->AcceptanceRate
         = (OperData->Parameters.Capacity
            - Period[OperData->Periods.PeriodCount - 1]) / ((double) MoreYears);
      }

   Results->YearClosed = OperData->Parameters.YearCurrent + MoreYears;

   if (MoreYears == 0) Results->AcceptanceRate = 0;

   GlobalUnlock(OperData->Periods.hPeriod);
}



void FillRefuseList(OPERDATASTRUCT *OperData, RESULTS *Results, double *Rate)
{
int Year;
int Index;
double *Period;
double Sum = 0;

   Period = (double *) GlobalLock(OperData->Periods.hPeriod);

   Year = Results->YearStart;
   for (Index = 0 ; Index < OperData->Periods.PeriodCount - 1 ; Index++, Year++)
      {
// Changed to Rate[index] to Rate[index - 1] to match older model

      Rate[Index] = Period[Index + 1] - Period[Index];
      Sum += Rate[Index];
      }

   for (; Index < Results->YearCount - 1 ; Index++, Year++)
      {
      if (Year == Results->YearClosed)
         Rate[Index] = OperData->Parameters.Capacity - Sum;
      else if (Year > Results->YearClosed)
         Rate[Index] = 0.0;
//       else Rate[Index] = Period[Index - 1] + Results->AcceptanceRate;
      else Rate[Index] = Results->AcceptanceRate;

      Sum += Rate[Index];
      }

   GlobalUnlock(OperData->Periods.hPeriod);
}



/* ========================================================================== */
/*  These functions use the ideal gas law to convert mass to volume and       */
/*                            volume to mass.
```

```
/*      All parameters are assumed to be in the correct units (MKS).       */
/* ========================================================================= */


double mass_to_volume(mass, mole_wt, temp, pressure)
double mass, mole_wt, temp, pressure;
{
   return(mass * (1000.0 * R) * temp / (mole_wt * pressure));
}



double volume_to_mass(volume, mole_wt, temp, pressure)
double volume, mole_wt, temp, pressure;
{
   return((mole_wt * pressure * volume) / (1000.0 * R * temp));
}


/* ========================================================================= */
/*  This is the actual model function.   It attempts to model the methane */
/*  emissions by assuming the generation is being emitted exponentially   */
/*  at different positions allow the emission curve by each subset of     */
/*  waste materials, depending on the time between the current year and   */
/*  the initial placement of the waste subset.  The emissions for each    */
/*  submass is calculated and summed to determine the total emission of   */
/*  the landfill as a whole.                                              */
/*                                                                        */
/*  Table of Variables                                                    */
/*                                                                        */
/*  k       The methane generation decay rate constant.                   */
/*  L0      The methane generation capacity constant.                     */
/*  refuse  The refuse acceptance rate list starting from the opening     */
/*          to the end calc year.                                         */
/*  i       The current year index (0 = 1 year after opening)             */
/* ========================================================================= */

double model_function(k, l0, refuse, i)
double l0, k, *refuse;
int i;
{
int j;
double sum = 0;

   for (j = 0 ; j <= i ; j++)
      sum += refuse[j] * exp(-k * (i - j));

   return(k * l0 * sum);
}
```

```
// ============================================================================
//                                  CLOSURE.C
// ============================================================================
// ============================================================================
//            Source code for displaying the ABOUT dialog box.
// ============================================================================
// ============================================================================
//                              Revision History
// ============================================================================
//  09/30/94    Radian(RLB)              Initial Revision
// ============================================================================

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

#include "landwin.h"
#include "str_ops.h"
#include "helpdef.h"


BOOL CALLBACK DialogProc(HWND hWndDlg, unsigned uiMsg, WORD wParam,
                         LONG lParam);


int GetClosureYear(HWND hWnd, HINSTANCE hInst)
{
FARPROC DlgProcInstance;
int ret_value;

   DlgProcInstance = MakeProcInstance(DialogProc, hInst);
   ret_value = DialogBox(hInst, "CLOSURE_DIALOG", hWnd, DlgProcInstance);
   FreeProcInstance(DlgProcInstance);

   return ret_value;
}


#pragma argsused

static BOOL CALLBACK DialogProc(HWND hWndDlg, unsigned uiMsg, WORD wParam,
                                LONG lParam)
{
char string[30];
HWND ActiveWindow;
HGLOBAL hOperData = NULL;
OPERDATASTRUCT *OperData;
int value;
BOOL ReloadFlag;

   switch (uiMsg)
      {
      case WM_INITDIALOG:
         CenterWindow(hWndDlg);

         ActiveWindow = SendMessage(RetrieveClientWindow(), WM_MDIGETACTIVE,
                                    0, 0);
         if (ActiveWindow != NULL)
            hOperData = SendMessage(ActiveWindow, AM_GETOPERHANDLE, 0, 0);

         if (hOperData != NULL)
            OperData = (OPERDATASTRUCT *) GlobalLock(hOperData);


         CustomizeControl(hWndDlg, IDC_VALUEEDIT, (FARPROC) IntNumberEditProc);
```



```
         if (OperData->Parameters.YearClosed == 0)
            {
            CheckRadioButton(hWndDlg, IDC_RADIOCALC, IDC_RADIOUSER, IDC_RADIOCAL
C);
            EnableWindow(GetDlgItem(hWndDlg, IDC_VALUEEDIT), FALSE);
            }
         else
            {
            CheckRadioButton(hWndDlg, IDC_RADIOCALC, IDC_RADIOUSER, IDC_RADIOUSE
R);
            EnableWindow(GetDlgItem(hWndDlg, IDC_VALUEEDIT), TRUE);
            }

         sprintf(string, "%4d", OperData->Parameters.YearClosed);
         SetDlgItemText(hWndDlg, IDC_VALUEEDIT, string);

         if (hOperData != NULL) GlobalUnlock(hOperData);
         return TRUE;

      case WM_COMMAND:
         switch (LOWORD(wParam))
            {
            case IDC_OK:
               ActiveWindow = SendMessage(RetrieveClientWindow(), WM_MDIGETACTIV
E,
                                          0, 0);
               if (ActiveWindow != NULL)
                  hOperData = SendMessage(ActiveWindow, AM_GETOPERHANDLE, 0, 0);

               if (hOperData != NULL)
                  OperData = (OPERDATASTRUCT *) GlobalLock(hOperData);


               if (IsDlgButtonChecked(hWndDlg, IDC_RADIOCALC))
                  OperData->Parameters.YearClosed = 0;
               else if (IsDlgButtonChecked(hWndDlg, IDC_RADIOUSER))
                  {
                  GetDlgItemText(hWndDlg, IDC_VALUEEDIT, string, sizeof(string))
;
                  value = atoi(string);

                  if (value < OperData->Parameters.YearCurrent)
                     OperData->Parameters.YearClosed = 0;
                  else OperData->Parameters.YearClosed = value;
                  }

               if (hOperData != NULL) GlobalUnlock(hOperData);

               EndDialog(hWndDlg, TRUE);
               return TRUE;

            case IDC_CANCEL:
               EndDialog(hWndDlg, FALSE);
               return TRUE;

            case IDC_HELP:
               WinHelp(hWndDlg, "LANDWIN.HLP", HELP_CONTEXT, ParametersClosure);
               return TRUE;

            case IDC_RADIOCALC:
            case IDC_RADIOUSER:
               if (IsDlgButtonChecked(hWndDlg, IDC_RADIOUSER))
                  EnableWindow(GetDlgItem(hWndDlg, IDC_VALUEEDIT), TRUE);
               else EnableWindow(GetDlgItem(hWndDlg, IDC_VALUEEDIT), FALSE);
               break;
```

```
            default:
               break;
            }
      }

   return FALSE;
}
```

```
// ============================================================================
//                                  CURRENT.C
// ============================================================================
// ============================================================================
//  Code for displaying a dialog box with the most current calculational
//  parameters.
// ----------------------------------------------------------------------------
// ============================================================================
//                              Revision History
// ----------------------------------------------------------------------------
//  09/30/94    Radian(RLB)             Initial Revision
// ============================================================================

#include <windows.h>
#include <stdio.h>

#include "landwin.h"

BOOL CALLBACK DialogProc(HWND hWndDlg, unsigned uiMsg, WORD wParam,
                         LONG lParam);

void DisplayCurrentParameters(HWND hWnd, HINSTANCE hInst)
{
FARPROC DlgProcInstance;
int ret_value;

   DlgProcInstance = MakeProcInstance(DialogProc, hInst);
   DialogBox(hInst, "CURRENT_PARAMS_DIALOG", hWnd, DlgProcInstance);
   FreeProcInstance(DlgProcInstance);
}


#pragma argsused

static BOOL CALLBACK DialogProc(HWND hWndDlg, unsigned uiMsg, WORD wParam,
                                LONG lParam)
{
HWND ActiveWindow;
HGLOBAL hOperData;
OPERDATASTRUCT *OperData;
char ValueString[30];

   switch (uiMsg)
      {
      case WM_INITDIALOG:
         CenterWindow(hWndDlg);

         ActiveWindow = SendMessage(RetrieveClientWindow(), WM_MDIGETACTIVE,
                                    0, 0);
         if (ActiveWindow != NULL)
            hOperData = SendMessage(ActiveWindow, AM_GETOPERHANDLE, 0, 0);

         if (hOperData != NULL)
            OperData = (OPERDATASTRUCT *) GlobalLock(hOperData);


         sprintf(ValueString, "%.4g", OperData->Model.k);
         SetDlgItemText(hWndDlg, IDC_KVALUE, (LPCSTR) ValueString);

         sprintf(ValueString, "%.4g", OperData->Model.10);
         SetDlgItemText(hWndDlg, IDC_LVALUE, (LPCSTR) ValueString);

         GlobalUnlock(hOperData);

         return TRUE;
```



```
      case WM_COMMAND:
         switch (LOWORD(wParam))
            {
            case IDC_OK:
               EndDialog(hWndDlg, TRUE);
               return TRUE;

            default:
               return FALSE;
            }

         default:
            return FALSE;
         }
}
```

```
// ========================================================================
//                                DOCLIP.C
// ========================================================================
// ========================================================================
//    Source code for saving the graphics output to the clipboard in
//    Windows Metafile format
// ========================================================================
// ========================================================================
//                            Revision History
// ========================================================================
//  09/30/94    Radian(RLB)             Initial Revision
// ========================================================================

#include <stdio.h>
#include <mem.h>
#include <windows.h>

#include "graphics.h"

#pragma argsused
void DoMetaFileToClipboard(HINSTANCE hInst, HWND hWnd)
{
HDC hdc;
RECT rect;
HGLOBAL hPtr;
HMETAFILE hmf;
HGLOBAL hmfpict;
METAFILEPICT *mfpict;

        hdc = CreateMetaFile(NULL);
        hPtr = (HGLOBAL) GetWindowWord(hWnd, 4);

        SetMapMode(hdc, MM_ANISOTROPIC);

        rect.top = 0;
        rect.left = 0;
        rect.right = 640;
        rect.bottom = 480;

   SetWindowExt(hdc, rect.right, rect.bottom);

        CalculateAxisDim(hdc, &rect, hPtr);
        WriteGraphTitle(hdc, &rect, hPtr);
        WriteAxisTitle(hdc, &rect, hPtr);
        DrawAxesLines(hdc, &rect, hPtr);
        DrawXAxisTicks(hdc, &rect, hPtr);
        DrawYAxisTicks(hdc, &rect, hPtr);
        DrawGraphSeries(hdc, &rect, hPtr);

        hmf = CloseMetaFile(hdc);

   hmfpict = GlobalAlloc(GHND, sizeof(METAFILEPICT));
   if (hmfpict == NULL) return;

   mfpict = (METAFILEPICT *) GlobalLock(hmfpict);
   if (mfpict == NULL) return;

   mfpict->mm = MM_ANISOTROPIC;
   mfpict->xExt = rect.right;
   mfpict->yExt = rect.bottom;
   mfpict->hMF = hmf;

   GlobalUnlock(hmfpict);

   if (OpenClipboard(hWnd))
      {
```



```
      EmptyClipboard();
      SetClipboardData(CF_METAFILEPICT, hmfpict);
      CloseClipboard();
      }
}
```

```
// ========================================================================
//                                DOPRINT.C
// ========================================================================
// ========================================================================
//                  Source for initiating a graphics print.
// ========================================================================
// ========================================================================
//                            Revision History
// ========================================================================
//  09/30/94    Radian(RLB)             Initial Revision
// ========================================================================
#include <stdio.h>
#include <mem.h>
#include <string.h>
#include <windows.h>
#include <print.h>
#include "commdlg.h"

#include "landwin.h"
#include "graphics.h"


typedef int (FAR *EXTDEVMODEFN)(HWND hWnd, HANDLE hDriver,
                                LPDEVMODE lpDevModeOut, LPSTR Device,
                                LPSTR Port, LPDEVMODE lpDevModeIn,
                                LPSTR Profile, WORD Mask);

typedef int (FAR *DEVMODEFN)(HWND hWnd, HANDLE hDriver,
                                LPSTR lpszDevice, LPSTR lpszPort);


typedef struct {
                                        char lpPrintDevice[40];
                                        char lpPrintDriver[20];
                                        char lpPrintPort[20];
                                        } PRINTERINFOSTRUCT;

static HANDLE hDM;
static PRINTERINFOSTRUCT pinfo;
static char DeviceDriver[80];

static int fPrtAbort;
static HWND AbortDlgBox;

void GetIndexedPrinter(PRINTERINFOSTRUCT *pinfo, int index);

#pragma argsused
void DoGraphPrintOutput(HINSTANCE hInst, HWND hWnd)
{
HGLOBAL hPtr;
GRAPHSTRUCT *pPtr;
HCURSOR hOldCursor;
RECT rect;
HDC hdc;
DLGPROC dlgproc;
int index, ret;
DOCINFO DocInfo;
DEVMODE *pDM = NULL;
void GetIndexedPrinter(PRINTERINFOSTRUCT *pinfo, int index);
BOOL CALLBACK PrintDlgProc(HWND hWndDlg, UINT msg, WPARAM wParam,

 LPARAM lParam);
void InitializeOrientation(int Orient);

// For Abort functions
FARPROC abortproc;
```



```
int FAR PASCAL AbortDlg(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam);
int CALLBACK AbortProc(HDC hdcPrinter, int Code);


        dlgproc = (DLGPROC) MakeProcInstance(PrintDlgProc, hInst);
        index = DialogBox(hInst, "DIALOG_PRINT", hWnd, dlgproc);
        FreeProcInstance((FARPROC) dlgproc);

   if (index == -1) return;

        GetIndexedPrinter(&pinfo, index);
   pDM = (DEVMODE *) GlobalLock(hDM);

        hdc = CreateDC(pinfo.lpPrintDriver,
                                                pinfo.lpPrintDevice,
                                                pinfo.lpPrintPort, (const void f
ar *) pDM);

   if (hDM != NULL)
      {
      GlobalUnlock(hDM);
      GlobalFree(hDM);
      }

        if (hdc != NULL)
           {
      fPrtAbort = FALSE;
        dlgproc = (DLGPROC) MakeProcInstance(AbortDlg, hInst);
      AbortDlgBox = CreateDialog(hInst, "PRTABORT_DIALOG", hWnd, dlgproc);

      abortproc = (FARPROC) MakeProcInstance(AbortProc, hInst);
      SetAbortProc(hdc, abortproc);

      EnableWindow(hWnd, FALSE);

        rect.top = 0;
           rect.left = 0;
                rect.right = GetDeviceCaps(hdc, HORZRES);
        rect.bottom = GetDeviceCaps(hdc, VERTRES);

      SetMapMode(hdc, MM_ANISOTROPIC);
      SetWindowExt(hdc, rect.right, rect.bottom);
      SetViewportExt(hdc, rect.right, rect.bottom);

        DocInfo.cbSize = sizeof(DocInfo);
           DocInfo.lpszDocName = "Graph";
                DocInfo.lpszOutput = NULL;

      hOldCursor = SetCursor(LoadCursor(NULL, IDC_WAIT));

        StartDoc(hdc, (DOCINFO FAR *) &DocInfo);
           StartPage(hdc);

      hPtr = (HGLOBAL) GetWindowLong(hWnd, 4);

      ClearMessages();

        CalculateAxisDim(hdc, &rect, hPtr);
           WriteGraphTitle(hdc, &rect, hPtr);
                WriteAxisTitle(hdc, &rect, hPtr);
      DrawAxesLines(hdc, &rect, hPtr);
           DrawXAxisTicks(hdc, &rect, hPtr);
                DrawYAxisTicks(hdc, &rect, hPtr);
      DrawGraphSeries(hdc, &rect, hPtr);

           EndPage(hdc);
```

```
                if (EndDoc(hdc) >= 0) DestroyWindow(AbortDlgBox);

        DeleteDC(hdc);

      SetCursor(hOldCursor);

      EnableWindow(hWnd, TRUE);
      FreeProcInstance(dlgproc);
      FreeProcInstance(abortproc);
      }
}



#pragma argsused
BOOL CALLBACK PrintDlgProc(HWND hWndDlg, UINT msg, WPARAM wParam,

 LPARAM lParam)
{
HWND hListBox;
HINSTANCE hDriver;
int index;
int PopulatePrintDialog(HWND hDialog);
LPFNDEVMODE lpfnExtDeviceMode;

        switch(msg)
                {
                case WM_INITDIALOG:
                        index = PopulatePrintDialog(hWndDlg);
        GetIndexedPrinter(&pinfo, index);

         strcpy(DeviceDriver, pinfo.lpPrintDriver);
        strcat(DeviceDriver, ".DRV");
        hDriver = LoadLibrary(DeviceDriver);

         lpfnExtDeviceMode
                        = (LPFNDEVMODE) GetProcAddress(hDriver, PROC_EXTDEVICEMO
DE);

         if (lpfnExtDeviceMode != NULL)
                        {
            int DMSize;
            DEVMODE *pDM;

            DMSize = (*lpfnExtDeviceMode)(hWndDlg, hDriver,
                                        NULL, pinfo.lpPrintDevice,
                                pinfo.lpPrintPort, NULL, NULL, 0);

            if (hDM != NULL) GlobalFree(hDM);

            hDM = (HGLOBAL) GlobalAlloc(GHND, DMSize);
            pDM = (DEVMODE *) GlobalLock(hDM);

            (*lpfnExtDeviceMode)(hWndDlg, hDriver, pDM, pinfo.lpPrintDevice,
                                pinfo.lpPrintPort, NULL, NULL,
                                DM_OUT_BUFFER);

            GlobalUnlock(hDM);
            }

         FreeLibrary(hDriver);
                        return TRUE;

                case WM_COMMAND:
                        switch (wParam)
                                {
```



```
                case LBN_DBLCLK:
                                        case IDC_PRINT:
                                                hListBox = GetDlgItem(hWndDlg, IDC_LISTB
OX1);
                                                index = (int) SendMessage(hListBox, LB_G
ETCURSEL, 0, 0);
                                                EndDialog(hWndDlg, index);
                                                return TRUE;

                                        case IDC_CANCEL:
                                                EndDialog(hWndDlg, -1);
                                                return TRUE;

                                        case IDC_SETUP:
                                                hListBox = GetDlgItem(hWndDlg, IDC_LISTB
OX1);
                                                index = (int) SendMessage(hListBox, LB_G
ETCURSEL, 0, 0);

                                                GetIndexedPrinter(&pinfo, index);

                                strcpy(DeviceDriver, pinfo.lpPrintDriver);
                                strcat(DeviceDriver, ".DRV");
                                                hDriver = LoadLibrary(DeviceDriver);

                                lpfnExtDeviceMode
                                                = (LPFNDEVMODE) GetProcAddress(hDriver,
PROC_EXTDEVICEMODE);

                                if (lpfnExtDeviceMode != NULL)
                                        {
                        DEVMODE *pDM;
                        int DMSize;

                        if (hDM != NULL) GlobalFree(hDM);

                        hDM = (HGLOBAL) GlobalAlloc(GHND, DMSize);
                        pDM = (DEVMODE *) GlobalLock(hDM);

                        (*lpfnExtDeviceMode)(hWndDlg, hDriver, pDM, pinfo.lpPrintDevic
e,
                                                pinfo.lpPrintPort, pDM, NULL,
                                                DM_IN_BUFFER | DM_IN_PROMPT | DM_OUT_BUFF
ER);

                        GlobalUnlock(hDM);
                        }

                FreeLibrary(hDriver);
                                                return TRUE;

                                        default:
                                                return FALSE;
                                        }

                 default:
                                return FALSE;
                 }
}



int PopulatePrintDialog(HWND hDialog)
```

```
{
HWND hListBox;
HLOCAL hBuffer;
LPSTR buffer, ptr;
PRINTERINFOSTRUCT pinfo;
char temp[80], defprinter[80];
int i, count = 0;
int ndefprt;
void ParsePrinterString(PRINTERINFOSTRUCT *ptr, char *buffer);

        hListBox = GetDlgItem(hDialog, IDC_LISTBOX1);

        GetProfileString("windows", "device", "", (LPSTR) defprinter, 80);
        *(strchr(defprinter, ',')) = '\0';

        hBuffer = LocalAlloc(LHND, 1024);
        if (hBuffer == NULL) return -1;

        buffer = (LPSTR) LocalLock(hBuffer);

        GetProfileString("printerPorts", NULL, "", buffer, 1024);

        ptr = buffer;
        for (ptr = buffer ; *ptr != '\0';)
                {
                while (*ptr != '\0') ptr++;

                ptr++;
                count++;
                }

        ptr = buffer;
        for (i = 0 ; i < count ; i++)
                {
                lstrcpy(pinfo.lpPrintDevice, ptr);
                while (*ptr != '\0') ptr++;
                ptr++;

                GetProfileString("printerPorts", pinfo.lpPrintDevice, "", temp,
80);

                ParsePrinterString(&pinfo, temp);

                sprintf(temp, "%s @ %s", pinfo.lpPrintDevice, pinfo.lpPrintPort)
;
                SendMessage(hListBox, LB_ADDSTRING, NULL, (LONG) (LPSTR)temp);

                if (! strcmp(defprinter, pinfo.lpPrintDevice))
                        {
                        SendMessage(hListBox, LB_SETCURSEL, i, 0);
                        ndefprt = i;
                        }
                }

        LocalUnlock(hBuffer);
        LocalFree(hBuffer);

        return ndefprt;
}



void GetIndexedPrinter(PRINTERINFOSTRUCT *pinfo, int index)
{
HLOCAL hBuffer;
```



```
LPSTR buffer, ptr;
char temp[80];
int i, count = 0;
void ParsePrinterString(PRINTERINFOSTRUCT *ptr, char *buffer);

        hBuffer = LocalAlloc(LHND, 1024);
        if (hBuffer == NULL) return;

        buffer = (LPSTR) LocalLock(hBuffer);

        GetProfileString("printerPorts", NULL, "", buffer, 1024);

        ptr = buffer;
        for (ptr = buffer ; *ptr != '\0';)
                {
                while (*ptr != '\0') ptr++;

                ptr++;
                count++;
                }

        ptr = buffer;
        for (i = 0 ; i < count ; i++)
                {
                lstrcpy(pinfo->lpPrintDevice, ptr);
                while (*ptr != '\0') ptr++;
                ptr++;

                if (i == index)
                        {
                        GetProfileString("printerPorts", pinfo->lpPrintDevice, "
", temp, 80);
                        ParsePrinterString(pinfo, temp);
                        break;
                        }
                }

        LocalUnlock(hBuffer);
        LocalFree(hBuffer);
}



void ParsePrinterString(PRINTERINFOSTRUCT *pinfo, char *buffer)
{
char *ptr = buffer;

        *(strchr(buffer, ',')) = '\0';
        lstrcpy((LPSTR) pinfo->lpPrintDriver, ptr);

        ptr += strlen(buffer) + 1;
        *(strchr(ptr, ',')) = '\0';
        lstrcpy((LPSTR) pinfo->lpPrintPort, ptr);
}



#pragma argsused
int FAR PASCAL AbortDlg(hWnd, msg, wParam, lParam)
HWND hWnd;
UINT msg;
WPARAM wParam;
```

```
LPARAM lParam;
{
    if (msg == WM_COMMAND) {
        if (wParam == IDC_CANCEL) {

            /* The user has canceled the print operation. */

            fPrtAbort = TRUE; /* global flag                */

            EndDialog(hWnd, wParam);
            return TRUE;
        }
    }
    else if (msg == WM_INITDIALOG) {

        /* Set the input focus for user input. */

        SetFocus(hWnd);
        return TRUE;
    }

   return FALSE;
}



#pragma argsused
int CALLBACK AbortProc(hdcPrinter, Code)
HDC hdcPrinter;   /* for multiple printer display contexts */
int Code;         /* for printing status                   */
{
    MSG msg;

/* Process the messages intended for the Abort dialog box. */

    while (PeekMessage((LPMSG) &msg,
           NULL, NULL, NULL, PM_REMOVE))
        if (!IsDialogMessage(AbortDlgBox,
                (LPMSG) &msg)) {
            TranslateMessage((LPMSG) &msg);
            DispatchMessage((LPMSG) &msg);
        }

/*
 * The fAbort argument is TRUE (return is FALSE)
 * if the user has canceled the print operation.
 */

    return (!fPrtAbort);
}
```

```
// ==========================================================================
//                                  DOWMF.C
// ==========================================================================
// ==========================================================================
//          Source for saving the graphic as a Windows Metafile
// ==========================================================================
// ==========================================================================
//                            Revision History
// ==========================================================================
//  09/30/94     Radian(RLB)              Initial Revision
// ==========================================================================
#include <stdio.h>
#include <mem.h>
#include <windows.h>
#include "commdlg.h"

#include "graphics.h"
#include "landres.h"

typedef struct {
                                        DWORD key;
                                        HANDLE hmf;
                                        RECT bbox;
                                        WORD inch;
                                        DWORD reserved;
                                        WORD checksum;
                                        } METAFILEHEADER;


void DoMetaFileOutput(HINSTANCE hInst, HWND hWnd)
{
HDC hdc;
RECT rect;
HGLOBAL hPtr;
HMETAFILE hmf;
char filename[256];
int GetWMFOutputFilename(HINSTANCE hInst, HWND hWnd, char *filename, int nSize);
void WriteWMFFile(HMETAFILE hmf, char *filename, RECT *rect);

        if (GetWMFOutputFilename(hInst, hWnd, filename, 255))
                {
                hdc = CreateMetaFile(NULL);
                hPtr = (HGLOBAL) GetWindowWord(hWnd, 4);

                SetMapMode(hdc, MM_ANISOTROPIC);

                rect.top = 0;
                rect.left = 0;
                rect.right = 32767;
                rect.bottom = 24575;

      SetWindowExt(hdc, rect.right, rect.bottom);

                CalculateAxisDim(hdc, &rect, hPtr);
                WriteGraphTitle(hdc, &rect, hPtr);
                WriteAxisTitle(hdc, &rect, hPtr);
                DrawAxesLines(hdc, &rect, hPtr);
                DrawXAxisTicks(hdc, &rect, hPtr);
                DrawYAxisTicks(hdc, &rect, hPtr);
                DrawGraphSeries(hdc, &rect, hPtr);

                hmf = CloseMetaFile(hdc);

                WriteWMFFile(hmf, filename, &rect);

                DeleteMetaFile(hmf);
```



```
                }
}


int GetWMFOutputFilename(HINSTANCE hInst, HWND hWnd, char *filename, int nSize)
{
OPENFILENAME ofn;
char szDirName[256];
char szFileTitle[256];
UINT  i, cbString;
char  chReplace;    /* string separator for szFilter */
char  szFilter[256];
int ret;

        *filename = '\0';
   GetSystemDirectory(szDirName, sizeof(szDirName));

        if ((cbString = LoadString(hInst, IDS_WMFFILTER,
                szFilter, sizeof(szFilter))) == 0) return 0;

        chReplace = szFilter[cbString - 1]; /* retrieve wildcard */

   for (i = 0; szFilter[i] != '\0'; i++)
                if (szFilter[i] == chReplace) szFilter[i] = '\0';

/* Set all structure members to zero. */

   memset(&ofn, 0, sizeof(OPENFILENAME));

   ofn.lStructSize = sizeof(OPENFILENAME);
   ofn.hwndOwner = hWnd;
   ofn.lpstrFilter = szFilter;
   ofn.nFilterIndex = 1;
        ofn.lpstrFile= filename;
   ofn.nMaxFile = nSize;
   ofn.lpstrFileTitle = szFileTitle;
   ofn.nMaxFileTitle = sizeof(szFileTitle);
   ofn.lpstrInitialDir = NULL;
   ofn.lpstrTitle = "Save Graph to File";
   ofn.Flags = OFN_PATHMUSTEXIST;
   ofn.lpstrDefExt = "WMF";

   ret = GetOpenFileName(&ofn);

        return ret;
}



void WriteWMFFile(HMETAFILE hmf, char *filename, RECT *rect)
{
HFILE hFile;
METAFILEHEADER mfh;
WORD *dwptr;
UINT count;
HGLOBAL hMfbits;
void _huge *data;
int i;

   hFile = _lcreat((LPCSTR) filename, 0);

        mfh.key = 0x9AC6CDD7L;
   mfh.hmf = 0;

   mfh.bbox.top = rect->top;
   mfh.bbox.left = rect->left;
```

```
        mfh.bbox.right = rect->right;
    mfh.bbox.bottom = rect->bottom;

    mfh.inch = 1000;
    mfh.reserved = 0;

    mfh.checksum = 0;
    dwptr = (WORD *) &mfh;
        for (i = 0 ; i < 10 ; i++) mfh.checksum ^= *(dwptr++);

    _lwrite(hFile, (void _huge *) &mfh, sizeof(METAFILEHEADER));

    hMfbits = GetMetaFileBits(hmf);
    count = (UINT) GlobalSize(hMfbits);

        data = GlobalLock(hMfbits);

    _lwrite(hFile, data, (UINT) count);

    GlobalUnlock(hMfbits);
    GlobalFree(hMfbits);

    _lclose(hFile);
}
```

```
// ==========================================================================
//                                ESTIMATE.C
// ==========================================================================
// ==========================================================================
//          Source for estimating waste from landfill dimensions
// ==========================================================================
// ==========================================================================
//                           Revision History
// ==========================================================================
//  09/30/94    Radian(RLB)              Initial Revision
// ==========================================================================

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

#include "landwin.h"
#include "str_ops.h"

BOOL CALLBACK DialogProc(HWND hWndDlg, unsigned uiMsg, WORD wParam,
                         LONG lParam);

static BOOL Continue = TRUE;


void RefuseEstimator(HWND hWnd, HINSTANCE hInst)
{
HWND hDlgBox;
FARPROC DlgProcInstance;
MSG Message;

   Continue = TRUE;

   DlgProcInstance = MakeProcInstance(DialogProc, hInst);
        hDlgBox = CreateDialog(hInst, "ESTIMATOR_DIALOG", hWnd, DlgProcInstance)
;

   while (GetMessage(&Message, NULL, 0, 0))
      {
      if (!IsDialogMessage(hDlgBox, &Message))
        {
        TranslateMessage(&Message);
        DispatchMessage(&Message);
        }

      if (! Continue) break;
      }

        FreeProcInstance(DlgProcInstance);
}


#pragma argsused

static BOOL CALLBACK DialogProc(HWND hWndDlg, unsigned uiMsg, WORD wParam,
                                LONG lParam)
{
char string[30];
double size, depth, estimate;
HWND ActiveWindow;
OPERDATASTRUCT *OperData;

   switch (uiMsg)
      {
      case WM_INITDIALOG:
```



```
         CenterWindow(hWndDlg);

         SetDlgItemText(hWndDlg, IDC_LANDSIZE, (LPCSTR) "0");
         CustomizeControl(hWndDlg, IDC_LANDSIZE, (FARPROC) FPNumberEditProc);
         SetDlgItemText(hWndDlg, IDC_LANDDEPTH, (LPCSTR) "0");
         CustomizeControl(hWndDlg, IDC_LANDDEPTH, (FARPROC) FPNumberEditProc);
         return TRUE;

      case WM_COMMAND:
         switch (LOWORD(wParam))
            {
            case IDC_OK:
               GetDlgItemText(hWndDlg, IDC_LANDSIZE, string, sizeof(string));
               size = strtod(string, NULL);
               GetDlgItemText(hWndDlg, IDC_LANDDEPTH, string, sizeof(string));
               depth = strtod(string, NULL);

               estimate = 1233.48 * size * depth * 0.636;

               sprintf(string, "%E", estimate);
               SetDlgItemText(hWndDlg, IDC_ESTIMATE, string);

               return TRUE;

            case IDC_CANCEL:
                                          DestroyWindow(hWndDlg);
               Continue = FALSE;
               return TRUE;

            case IDC_HELP:
               return TRUE;

            default:
               break;
            }
      }

   return FALSE;
}
```

```
// ==========================================================================
//                              GRAPHICS.C
// ==========================================================================
// ==========================================================================
//              Source for creating and depicting a graphic
// ==========================================================================
// ==========================================================================
//                            Revision History
// ==========================================================================
//  09/30/94    Radian(RLB)             Initial Revision
// ==========================================================================
#include <stdio.h>
#include <string.h>
#include <mem.h>
#include <math.h>
#include <windows.h>
#include "commdlg.h"

#include "landwin.h"
#include "results.h"
#include "graphics.h"

#define PercentToPixelX(rect, percent) \
        ((int)((((double) (rect->right)) * ((double) (percent))) / 100.0))
#define PercentToPixelY(rect, percent) \
        ((int)((((double) (rect->bottom)) * ((double) (percent))) / 100.0))
#define PixelToPercentX(rect, pixel) \
        (((double) (pixel) * 100) / (double) (rect->right))
#define PixelToPercentY(rect, pixel) \
        (((double) (pixel) * 100) / (double) (rect->bottom))


HWND CreateGraphClientWindow(char *title, int x, int y, int width, int height,
                             DWORD style)
{
MDICREATESTRUCT MDIcs;
HWND hWnd;

/* Set up the MDI Structure to create the client window.  lParam contains */
/* an id that is stored in the window that indicates the type of MDI */
/* window this is and which WndProc needs to be invoked. */

   MDIcs.szClass = "GRAPHWINDOW";
   MDIcs.szTitle = title;
   MDIcs.hOwner = RetrieveAppInstance();
   MDIcs.x = x;
   MDIcs.y = y;
   MDIcs.cx = width;
   MDIcs.cy = height;
   MDIcs.style = style;
   MDIcs.lParam = 0;

   hWnd = SendMessage(RetrieveClientWindow(), WM_MDICREATE, 0, (LPARAM) &MDIcs);

   return hWnd;
}



long FAR PASCAL GraphWndProc(HWND hWnd, WORD iMessage, WPARAM wParam, LPARAM lPa
ram)
{
HDC hdc;
PAINTSTRUCT lpps;
RECT rect;
HGLOBAL hPtr, hResults;
```



```
RESULTS *Results;
HMENU hmenu;
DLGPROC dlgproc;
void DoGraphPrintOutput(HINSTANCE hInst, HWND hWnd);
void DoMetaFileOutput(HINSTANCE hInst, HWND hWnd);
void DoMetaFileToClipboard(HINSTANCE hInst, HWND hWnd);
void SetGraphDefaults(HWND hWnd);
void FormatLandwinGraph(HWND hWnd, RESULTS *Results);
void DestroySeriesStruct(HGLOBAL hGraph);
int GetOutputFilename(HWND hWnd, char *filename, int nSize);

     switch (iMessage)
        {
        case WM_CREATE:
           hPtr = GlobalAlloc(GHND, sizeof(GRAPHSTRUCT));
           SetWindowLong(hWnd, 4, (long) hPtr);
           SetGraphDefaults(hWnd);

           hmenu = GetSystemMenu(hWnd, FALSE);
           AppendMenu(hmenu, MF_SEPARATOR, 0, (LPSTR) NULL);
           AppendMenu(hmenu, MF_STRING, SYSMENU_SAVE, "&Save ...");
           AppendMenu(hmenu, MF_STRING, SYSMENU_PRINT, "&Print ...");
           AppendMenu(hmenu, MF_SEPARATOR, 0, (LPSTR) NULL);
           AppendMenu(hmenu, MF_STRING, SYSMENU_COPY, "&Copy");
           AppendMenu(hmenu, MF_SEPARATOR, 0, (LPSTR) NULL);
           AppendMenu(hmenu, MF_STRING, SYSMENU_ABOUT, "&About ...");

           SetWindowLong(hWnd, 8, (long) NULL);

           return TRUE;

        case WM_MDIACTIVATE:
           SendMessage(RetrieveFrameWindow(), AM_WINDOWACTIVE, 0, 0);
           return TRUE;

        case AM_GETOPERHANDLE:
           return (long) GetWindowLong(hWnd, 0);

        case AM_VALUESCHANGED:
           hPtr = (HGLOBAL) GetWindowLong(hWnd, 8);

           if (hPtr != NULL)
              {
              Results = (RESULTS *) GlobalLock(hPtr);

              if (ValidateData())
                 {
                 hResults = CalculationModule(Results->AirToxicType);
                 if (hResults == NULL)
                    {
                    GlobalUnlock(hPtr);
                    return TRUE;
                    }
                 }
              else
                 {
                 GlobalUnlock(hPtr);
                 return TRUE;
                 }

              if (Results->hRate != NULL) GlobalFree(Results->hRate);

              GlobalUnlock(hPtr);
              GlobalFree(hPtr);

              SetWindowLong(hWnd, 8, hResults);
```

```
            InvalidateRect(hWnd, NULL, TRUE);
            }

         return TRUE;

      case WM_PAINT:
         BeginPaint(hWnd, &lpps);

         if ((hPtr = (HGLOBAL) GetWindowLong(hWnd, 8)) != NULL)
            {
            Results = (RESULTS *) GlobalLock(hPtr);
            FormatLandwinGraph(hWnd, Results);
            GlobalUnlock(hPtr);

            GetClientRect(hWnd, &rect);
            hdc = lpps.hdc;

            SetMapMode(hdc, MM_ANISOTROPIC);
            SetWindowExt(hdc, rect.right, rect.bottom);
            SetViewportExt(hdc, rect.right, rect.bottom);

            hPtr = (HGLOBAL) GetWindowLong(hWnd, 4);

            CalculateAxisDim(hdc, &rect, hPtr);
            WriteGraphTitle(hdc, &rect, hPtr);
            WriteAxisTitle(hdc, &rect, hPtr);
            DrawAxesLines(hdc, &rect, hPtr);
                        DrawXAxisTicks(hdc, &rect, hPtr);
            DrawYAxisTicks(hdc, &rect, hPtr);
            DrawGraphSeries(hdc, &rect, hPtr);
            }

         EndPaint(hWnd, &lpps);
         return TRUE;

      case WM_DESTROY:
         hPtr = (HGLOBAL) GetWindowLong(hWnd, 4);
         DestroySeriesStruct(hPtr);
         if (hPtr != NULL) GlobalFree(hPtr);

         hPtr = (HGLOBAL) GetWindowLong(hWnd, 8);
         if (hPtr != NULL)
            {
            Results = (RESULTS *) GlobalLock(hPtr);
            if (Results->hRate != NULL) GlobalFree(Results->hRate);
            GlobalUnlock(hPtr);
            GlobalFree(hPtr);
            }

         PostMessage(RetrieveFrameWindow(), AM_WINDOWACTIVE, 0, 0L);
         break;

      case AM_SAVE:
         DoMetaFileOutput(RetrieveAppInstance(), hWnd);
         return TRUE;

               case AM_PRINT:
                     DoGraphPrintOutput(RetrieveAppInstance(), hWnd);
                     return TRUE;

      case AM_COPY:
         DoMetaFileToClipboard(RetrieveAppInstance(), hWnd);
         return TRUE;

      case WM_SYSCOMMAND:
         {
```



```
            switch (wParam)
               {
               case SYSMENU_ABOUT:
                  dlgproc = (DLGPROC) MakeProcInstance(AboutDlgProc, RetrieveApp
Instance());
                  DialogBox(RetrieveAppInstance(), "DIALOG_1", hWnd, dlgproc);
                  FreeProcInstance((FARPROC) dlgproc);
                  return TRUE;

               case SYSMENU_SAVE:
                                                DoMetaFileOutput(RetrieveAppInst
ance(), hWnd);
                  return TRUE;

                                    case SYSMENU_PRINT:
                                                DoGraphPrintOutput(RetrieveAppIn
stance(), hWnd);
                                                return TRUE;

               case SYSMENU_COPY:
                  DoMetaFileToClipboard(RetrieveAppInstance(), hWnd);
                  return TRUE;

               default:
                  break;
               }
            }
         }

      return(DefMDIChildProc(hWnd, iMessage, wParam, lParam));
}


void FormatLandwinGraph(HWND hWnd, RESULTS *Results)
{
HGLOBAL hPoints;
HGLOBAL hSeries;
POINTSTRUCT *Points;
SERIESSTRUCT *Series;
char title[80];
int i;
double yMin = 0, yMax = 0;
double *Value;
void FormatAxis(HWND hWNd, int type, double Min, double Max);
HGLOBAL GetGraphSeries(HWND hWnd);

   if ((hSeries = GetGraphSeries(hWnd)) != NULL)
      {
      Series = (SERIESSTRUCT *) GlobalLock(hSeries);
      GlobalFree(Series->hPoints);
      GlobalUnlock(hSeries);
      GlobalFree(hSeries);

      SetGraphSeries(hWnd, NULL, 0);
      }

   sprintf(title, "Projected %s Emissions", Results->Name);
   SetGraphTitles(hWnd, title, "");

   SetAxisTitle(hWnd, X_AXIS, "Year");
   SetAxisTitle(hWnd, Y_AXIS, "Emissions (Mg)");

   FormatAxis(hWnd, X_AXIS, (double) Results->YearStart,
                     (double) Results->YearStart + Results->YearCount);
```

```
   Value = ((double *) GlobalLock(Results->hRate))
                     + Results->YearCount;

   hPoints = GlobalAlloc(GHND, Results->YearCount * sizeof(POINTSTRUCT));
   if (hPoints == NULL) return;

   Points = (POINTSTRUCT *) GlobalLock(hPoints);

   for (i = 0 ; i < Results->YearCount ; i++)
      {
      yMin = (yMin < Value[i]) ? yMin : Value[i];
      yMax = (yMax > Value[i]) ? yMax : Value[i];

      Points[i].x = i + Results->YearStart;
      Points[i].y = Value[i];
      }

   GlobalUnlock(hPoints);

   FormatAxis(hWnd, Y_AXIS, 0.0, yMax);

   SetAxisLabelFormat(hWnd, X_AXIS, "%.0f");
   SetAxisLabelFormat(hWnd, Y_AXIS, "%.2E");

   hSeries = GlobalAlloc(GHND, sizeof(SERIESSTRUCT));
   Series = (SERIESSTRUCT *) GlobalLock(hSeries);

   Series->lineStyle = PS_SOLID;
   Series->lineColor = RGB(0, 0, 0);
   Series->pointCount = Results->YearCount;
   Series->hPoints = hPoints;

   GlobalUnlock(hSeries);

   SetGraphSeries(hWnd, hSeries, 1);
}



void DestroySeriesStruct(HGLOBAL hGraph)
{
int i;
GRAPHSTRUCT *ptr = (GRAPHSTRUCT *) GlobalLock(hGraph);
SERIESSTRUCT *Series;

   if (ptr->hSeries != NULL)
      {
      Series = (SERIESSTRUCT *) GlobalLock(ptr->hSeries);
      for (i = 0 ; i < ptr->seriesCount ; i++)
         if (Series->hPoints != NULL) GlobalFree(Series->hPoints);

      GlobalUnlock(ptr->hSeries);
      GlobalFree(ptr->hSeries);
      }
}


void FormatAxis(HWND hWnd, int type, double Min, double Max)
{
double range, base;
double MajorTicks, MinorTicks;
double AxisMin, AxisMax;

   range = Max - Min;

   base = range / pow(10.0, floor(log10(range)));
```



```
   if (base < 1.5) base = 1.0;
   if ((base >= 1.5) && (base < 3.0)) base = 2.0;
   if (base >= 3.0) base = 5.0;

   MajorTicks = base * pow(10.0, floor(log10(range)) - 1);
   MinorTicks = base * pow(10.0, floor(log10(range)) - 2);

   AxisMin = floor(Min / MajorTicks) * MajorTicks;
        AxisMax =  ceil(Max / MajorTicks) * MajorTicks;

   SetAxisScale(hWnd, type, AxisMin, AxisMax);
   SetAxisTicks(hWnd, type, MinorTicks, 10);
}



void SetGraphDefaults(HWND hWnd)
{
   SetGraphFont(hWnd, "Times New Roman");
   SetGraphTitles(hWnd, "Y versus X", "");
   SetAxisTitle(hWnd, X_AXIS, "X");
   SetAxisTitle(hWnd, Y_AXIS, "Y");
}



void SetGraphFont(HWND hWnd, char *fontName)
{
HGLOBAL hPtr = (HGLOBAL) GetWindowLong(hWnd, 4);
GRAPHSTRUCT *ptr;

   if (hPtr == NULL) return;

   ptr = (GRAPHSTRUCT *) GlobalLock(hPtr);

   strncpy(ptr->titles.fontName, fontName, sizeof(ptr->titles.fontName));

   GlobalUnlock(hPtr);
}



void SetGraphTitles(HWND hWnd, char *PrimaryTitle,
                    char *SecondaryTitle)
{
HGLOBAL hPtr = (HGLOBAL) GetWindowLong(hWnd, 4);
GRAPHSTRUCT *ptr;

   if (hPtr == NULL) return;

   ptr = (GRAPHSTRUCT *) GlobalLock(hPtr);

   strncpy(ptr->titles.primaryTitle, PrimaryTitle,
                sizeof(ptr->titles.primaryTitle));
   strncpy(ptr->titles.secondaryTitle, SecondaryTitle,
                sizeof(ptr->titles.secondaryTitle));

   GlobalUnlock(hPtr);
}


void SetGraphSeries(HWND hWnd, HGLOBAL hSeries, int count)
{
HGLOBAL hPtr = (HGLOBAL) GetWindowLong(hWnd, 4);
```

```
GRAPHSTRUCT *ptr;

   if (hPtr == NULL) return;

   ptr = (GRAPHSTRUCT *) GlobalLock(hPtr);

   ptr->hSeries = hSeries;
   ptr->seriesCount = count;

   GlobalUnlock(hPtr);
}



HGLOBAL GetGraphSeries(HWND hWnd)
{
HGLOBAL hPtr = (HGLOBAL) GetWindowLong(hWnd, 4);
HGLOBAL hSeries;
GRAPHSTRUCT *ptr;

   if (hPtr == NULL) return NULL;

   ptr = (GRAPHSTRUCT *) GlobalLock(hPtr);

   hSeries = ptr->hSeries;

   GlobalUnlock(hPtr);

   return hSeries;
}



void SetAxisTitle(HWND hWnd, int axis, char *title)
{
HGLOBAL hPtr = (HGLOBAL) GetWindowLong(hWnd, 4);
GRAPHSTRUCT *ptr;

   if (hPtr == NULL) return;

   ptr = (GRAPHSTRUCT *) GlobalLock(hPtr);

   if (axis == X_AXIS)
      strncpy(ptr->xAxis.axisTitle, title, sizeof(ptr->xAxis.axisTitle));
   else
      strncpy(ptr->yAxis.axisTitle, title, sizeof(ptr->yAxis.axisTitle));

   GlobalUnlock(hPtr);
}


void SetAxisLabelFormat(HWND hWnd, int axis, char *format)
{
HGLOBAL hPtr = (HGLOBAL) GetWindowLong(hWnd, 4);
GRAPHSTRUCT *ptr;

   if (hPtr == NULL) return;

   ptr = (GRAPHSTRUCT *) GlobalLock(hPtr);

   if (axis == X_AXIS)
      strncpy(ptr->xAxis.labelFormat, format, sizeof(ptr->xAxis.labelFormat));
   else
```



```
      strncpy(ptr->yAxis.labelFormat, format, sizeof(ptr->yAxis.labelFormat));

   GlobalUnlock(hPtr);
}



void SetAxisScale(HWND hWnd, int axis, double min, double max)
{
HGLOBAL hPtr = (HGLOBAL) GetWindowLong(hWnd, 4);
GRAPHSTRUCT *ptr;

   if (hPtr == NULL) return;

   ptr = (GRAPHSTRUCT *) GlobalLock(hPtr);

   if (axis == X_AXIS)
      {
      ptr->xAxis.min = min;
      ptr->xAxis.max = max;
      }
   else
      {
      ptr->yAxis.min = min;
      ptr->yAxis.max = max;
      }

   GlobalUnlock(hPtr);
}



void SetAxisTicks(HWND hWnd, int axis, double minorInterval,
                  int majorInterval)
{
HGLOBAL hPtr = (HGLOBAL) GetWindowLong(hWnd, 4);
GRAPHSTRUCT *ptr;

   if (hPtr == NULL) return;

   ptr = (GRAPHSTRUCT *) GlobalLock(hPtr);

   if (axis == X_AXIS)
      {
      ptr->xAxis.majorInterval = majorInterval;
      ptr->xAxis.minorInterval = minorInterval;
      }
   else
      {
      ptr->yAxis.minorInterval = minorInterval;
      ptr->yAxis.majorInterval = majorInterval;
      }

   GlobalUnlock(hPtr);
}



int WriteGraphTitle(HDC hdc, RECT *rect, HGLOBAL hPtr)
{
DWORD TextExtent;
HFONT hfnt, hprevfnt;
HLOCAL hlf = LocalAlloc(LPTR, sizeof(LOGFONT));
GRAPHSTRUCT *ptr;
```

```
LOGFONT *plf;
int posx, posy;

   ptr = (GRAPHSTRUCT *) GlobalLock(hPtr);

   plf = (LOGFONT *) LocalLock(hlf);
   if (plf == NULL) return FALSE;

   plf->lfHeight = PercentToPixelY(rect, 7.0);
   plf->lfWidth = PercentToPixelX(rect, 1.5);
   lstrcpy(plf->lfFaceName, ptr->titles.fontName);

   hfnt = CreateFontIndirect(plf);
   hprevfnt = SelectObject(hdc, hfnt);

   SetTextAlign(hdc, TA_CENTER | TA_TOP | TA_NOUPDATECP);

   posx = PercentToPixelX(rect, 50.0);
   posy = PercentToPixelY(rect, 2.0);

   TextOut(hdc, posx, posy, ptr->titles.primaryTitle,
           lstrlen(ptr->titles.primaryTitle));

   plf->lfHeight = PercentToPixelY(rect, 5.0);
   plf->lfWidth = PercentToPixelX(rect, 1.3);
   lstrcpy(plf->lfFaceName, ptr->titles.fontName);

   posy = PercentToPixelY(rect, 9.0);

   SelectObject(hdc, hprevfnt);
   DeleteObject(hfnt);

   hfnt = CreateFontIndirect(plf);
   hprevfnt = SelectObject(hdc, hfnt);

   TextOut(hdc, posx, posy, ptr->titles.secondaryTitle,
           lstrlen(ptr->titles.secondaryTitle));

   GlobalUnlock(hPtr);

   LocalUnlock(hlf);
        LocalFree(hlf);
   SelectObject(hdc, hprevfnt);

        if (DeleteObject(hfnt)) return TRUE;
   else return FALSE;
}


int WriteAxisTitle(HDC hdc, RECT *rect, HGLOBAL hPtr)
{
DWORD TextExtent;
HFONT hfnt, hprevfnt;
HLOCAL hlf = LocalAlloc(LPTR, sizeof(LOGFONT));
GRAPHSTRUCT *ptr;
LOGFONT *plf;
int posx, posy;

   SetTextAlign(hdc, TA_LEFT | TA_TOP | TA_NOUPDATECP);

   p:    GRAPHSTRUCT *) GlobalLock(hPtr);

   plf   LOGFONT *) LocalLock(hlf);
   if (plf == NULL) return FALSE;

   plf->lfHeight = PercentToPixelY(rect, 5.0);
```



```
   plf->lfWidth = PercentToPixelX(rect, 1.3);
   lstrcpy(plf->lfFaceName, ptr->titles.fontName);

   hfnt = CreateFontIndirect(plf);
        hprevfnt = SelectObject(hdc, hfnt);

   TextExtent
      = GetTextExtent(hdc, ptr->xAxis.axisTitle, strlen(ptr->xAxis.axisTitle));

   if (TextExtent == 0)
      TextExtent = MAKELONG(plf->lfWidth * strlen(ptr->xAxis.axisTitle),
                            plf->lfHeight);

        posx = PercentToPixelX(rect,
                        ptr->xAxis.axisStart
                              + (ptr->xAxis.axisEnd - ptr->xAxis.axisStart) /
2)
                - LOWORD(TextExtent) / 2;
        posy = PercentToPixelY(rect, 90.0);

   TextOut(hdc, posx, posy, ptr->xAxis.axisTitle,
           lstrlen(ptr->xAxis.axisTitle));

   SelectObject(hdc, hprevfnt);

   DeleteObject(hfnt);

   plf->lfHeight = PercentToPixelY(rect, 5.0);
   plf->lfWidth  = PercentToPixelX(rect, 1.3);
   lstrcpy(plf->lfFaceName, ptr->titles.fontName);
   plf->lfEscapement = 2700;

   hfnt = CreateFontIndirect(plf);
   hprevfnt = SelectObject(hdc, hfnt);

   TextExtent
      = GetTextExtent(hdc, ptr->yAxis.axisTitle, strlen(ptr->yA    sTitle));

   if (TextExtent == 0)
      TextExtent = MAKELONG(plf->lfWidth * strlen(ptr->yAxis.axisTitle),
                            plf->lfHeight);

        posx = PercentToPixelX(rect, 7.0) - HIWORD(TextExtent) / 2;  // Was 5.0
then 7.0
   posy = PercentToPixelY(rect, 50.0) - LOWORD(TextExtent) / 2;

   TextOut(hdc, posx, posy, ptr->yAxis.axisTitle,
           lstrlen(ptr->yAxis.axisTitle));

   GlobalUnlock(hPtr);

   LocalUnlock(hlf);
        LocalFree(hlf);
        SelectObject(hdc, hprevfnt);

        if (DeleteObject(hfnt)) return TRUE;
        else return FALSE;
}

int CalculateAxisDim(HDC hdc, RECT *rect, HGLOBAL hPtr)
{
char label[30];
HFONT hfnt, hprevfnt;
GRAPHSTRUCT *ptr;
HLOCAL hlf = LocalAlloc(LPTR, sizeof(LOGFONT));
LOGFONT *plf;
```

```
int max_extent = 0, extent;
int majorTicks = 0;
double minorTicks;

   ptr = (GRAPHSTRUCT *) GlobalLock(hPtr);

        plf = (LOGFONT *) LocalLock(hlf);
        if (plf == NULL) return FALSE;

        plf->lfHeight = PercentToPixelY(rect, 5.0);
        plf->lfWidth = PercentToPixelX(rect, 0.8);

        lstrcpy(plf->lfFaceName, ptr->titles.fontName);

        hfnt = CreateFontIndirect(plf);
        hprevfnt = SelectObject(hdc, hfnt);

        SetTextAlign(hdc, TA_RIGHT | TA_TOP );

        for (minorTicks = ptr->yAxis.min ;
                  minorTicks <= ptr->yAxis.max;
                  minorTicks += ptr->yAxis.minorInterval)
                {
                if ((majorTicks == ptr->yAxis.majorInterval) || (majorTicks == 0
))
                        {
                        if (ptr->yAxis.labelFormat != NULL)
                                sprintf(label, ptr->yAxis.labelFormat, minorTick
s);
                        else
                                sprintf(label, "%f", minorTicks);

                        extent = lstrlen(label) * PercentToPixelX(rect, 1.3);

                        max_extent = max_extent < extent ? extent : max_extent;


                        majorTicks = 1;
                        }
                else majorTicks++;
                }

        ptr->xAxis.axisStart = 10.0
                                        + PixelToPercentX(rect, max_extent);

        ptr->xAxis.axisEnd = 96.0;

   ptr->yAxis.axisStart = 19.0;
   ptr->yAxis.axisEnd = 80.0;

   GlobalUnlock(hPtr);

   LocalUnlock(hlf);
   LocalFree(hlf);
   SelectObject(hdc, hprevfnt);

   if (DeleteObject(hfnt)) return TRUE;
   else return FALSE;
}



int DrawAxesLines(HDC hdc, RECT *rect, HGLOBAL hPtr)
{
GRAPHSTRUCT *ptr;
```



```
   ptr = (GRAPHSTRUCT *) GlobalLock(hPtr);
   if (ptr == NULL) return FALSE;

   MoveToEx(hdc, PercentToPixelX(rect, ptr->xAxis.axisStart),
                 PercentToPixelY(rect, ptr->yAxis.axisEnd), NULL);
   LineTo(hdc, PercentToPixelX(rect, ptr->xAxis.axisEnd),
               PercentToPixelY(rect, ptr->yAxis.axisEnd));
        MoveToEx(hdc, PercentToPixelX(rect, ptr->xAxis.axisStart),
                 PercentToPixelY(rect, ptr->yAxis.axisEnd), NULL);
   LineTo(hdc, PercentToPixelX(rect, ptr->xAxis.axisStart),
                                     PercentToPixelY(rect, ptr->yAxis.axisSta
rt));

   GlobalUnlock(hPtr);

   return TRUE;
}



int DrawXAxisTicks(HDC hdc, RECT *rect, HGLOBAL hPtr)
{
char label[30];
HFONT hfnt, hprevfnt;
GRAPHSTRUCT *ptr;
HLOCAL hlf = LocalAlloc(LPTR, sizeof(LOGFONT));
LOGFONT *plf;
double slope, intercept;
double value;
int posx, posy;
int majorTicks = 0;
double minorTicks;

   ptr = (GRAPHSTRUCT *) GlobalLock(hPtr);
   if (ptr == NULL) return FALSE;

        plf = (LOGFONT *) LocalLock(hlf);
   if (plf == NULL) return FALSE;

   plf->lfHeight = PercentToPixelY(rect, 3.0);
   plf->lfWidth = PercentToPixelX(rect, .8); //Was 1.3
        lstrcpy(plf->lfFaceName, ptr->titles.fontName);

   hfnt = CreateFontIndirect(plf);
   hprevfnt = SelectObject(hdc, hfnt);

   SetTextAlign(hdc, TA_CENTER | TA_BASELINE | TA_NOUPDATECP);

   slope = (ptr->xAxis.axisEnd - ptr->xAxis.axisStart)
                / (ptr->xAxis.max - ptr->xAxis.min);
   intercept = ptr->xAxis.axisStart;

   for (minorTicks = ptr->xAxis.min ;
        minorTicks <= ptr->xAxis.max;
                  minorTicks += ptr->xAxis.minorInterval)
      {
      value = slope * (minorTicks - ptr->xAxis.min) + intercept;

      MoveToEx(hdc, PercentToPixelX(rect, value),
                    PercentToPixelY(rect, ptr->yAxis.axisEnd), NULL);
      LineTo(hdc, PercentToPixelX(rect, value),
                  PercentToPixelY(rect, ptr->yAxis.axisEnd + 1.0));

      if ((majorTicks == ptr->xAxis.majorInterval) || (majorTicks == 0))
         {
         LineTo(hdc, PercentToPixelX(rect, value),
```

```
                                                    PercentToPixelY(rect, pt
r->yAxis.axisEnd + 2.0));

         if (ptr->xAxis.labelFormat != NULL)
            sprintf(label, ptr->xAxis.labelFormat, minorTicks);
         else
            sprintf(label, "%f", minorTicks);

                        posx = PercentToPixelX(rect, value);
         posy = PercentToPixelY(rect, ptr->yAxis.axisEnd + 4.0);


         TextOut(hdc, posx, posy, label, lstrlen(label));

         majorTicks = 1;
         }
      else majorTicks++;
      }

   GlobalUnlock(hPtr);

   LocalUnlock(hlf);
   LocalFree(hlf);
   SelectObject(hdc, hprevfnt);

   if (DeleteObject(hfnt)) return TRUE;
   else return FALSE;
}



int DrawYAxisTicks(HDC hdc, RECT *rect, HGLOBAL hPtr)
{
char label[30];
DWORD TextExtent;
HFONT hfnt, hprevfnt;
GRAPHSTRUCT *ptr;
HLOCAL hlf = LocalAlloc(LPTR, sizeof(LOGFONT));
LOGFONT *plf;
double slope, intercept;
double value;
int majorTicks = 0;
int posx, posy;
double minorTicks;

   ptr = (GRAPHSTRUCT *) GlobalLock(hPtr);

        plf = (LOGFONT *) LocalLock(hlf);
   if (plf == NULL) return FALSE;

        plf->lfHeight = PercentToPixelY(rect, 3.0);
        plf->lfWidth = PercentToPixelX(rect, .6);  // Was 1.3
   lstrcpy(plf->lfFaceName, ptr->titles.fontName);

   hfnt = CreateFontIndirect(plf);
   hprevfnt = SelectObject(hdc, hfnt);

   SetTextAlign(hdc, TA_RIGHT | TA_TOP | TA_NOUPDATECP);

        slope = (ptr->yAxis.axisStart - ptr->yAxis.axisEnd)
                / (ptr->yAxis.max - ptr->yAxis.min);
   intercept = ptr->yAxis.axisEnd;

   for (minorTicks = ptr->yAxis.min ;
        minorTicks <= ptr->yAxis.max;
        minorTicks += ptr->yAxis.minorInterval)
```



```
      {
      value = slope * (minorTicks - ptr->yAxis.min) + intercept;

      MoveToEx(hdc, PercentToPixelX(rect, ptr->xAxis.axisStart),
                                                    PercentToPixelY(rect,
NULL);
      LineTo(hdc, PercentToPixelX(rect, ptr->xAxis.axisStart - 1.0),
                  PercentToPixelY(rect, value));

      if ((majorTicks == ptr->yAxis.majorInterval) || (majorTicks == 0))
                  {
         LineTo(hdc, PercentToPixelX(rect, ptr->xAxis.axisStart - 2.0),
                     PercentToPixelY(rect, value));

         if (ptr->yAxis.labelFormat != NULL)
                                 sprintf(label, ptr->yAxis.labelFormat, m
s);
         else
            sprintf(label, "%f", minorTicks);

         TextExtent = GetTextExtent(hdc, label, strlen(label));

         posx = PercentToPixelX(rect, ptr->xAxis.axisStart - 3.0);
                        posy = PercentToPixelY(rect, value)
                                                    - HIWORD(TextE
 2;


         TextOut(hdc, posx, posy, label, lstrlen(label));

         majorTicks = 1;
         }
      else majorTicks++;
      }

   GlobalUnlock(hPtr);

   LocalUnlock(hlf);
   LocalFree(hlf);
   SelectObject(hdc, hprevfnt);

   if (DeleteObject(hfnt)) return TRUE;
   else return FALSE;
}


int DrawGraphSeries(HDC hdc, RECT *rect, HGLOBAL hPtr)
{
HRGN ClipRegion;
GRAPHSTRUCT *ptr;
POINTSTRUCT *points;
SERIESSTRUCT *series;
int pointCount;
HPEN hpen, hprevpen;
double slope_x, intercept_x;
double slope_y, intercept_y;
double percent_x, percent_y;
int i, j;

   ptr = (GRAPHSTRUCT *) GlobalLock(hPtr);
   if (ptr == NULL) return FALSE;

   series = (SERIESSTRUCT *) GlobalLock(ptr->hSeries);
        if (series == NULL) return FALSE;

        IntersectClipRect(hdc, PercentToPixelX(rect, ptr->xAxis.axisStar
                                                    Percen
```

```
Y(rect, ptr->yAxis.axisStart),
                                                        PercentToPixel
X(rect, ptr->xAxis.axisEnd),
                                                        PercentToPixel
Y(rect, ptr->yAxis.axisEnd));

        slope_x = (ptr->xAxis.axisEnd - ptr->xAxis.axisStart)
                                                / (ptr->xAxis.max - ptr->xAxis.m
in);
   intercept_x = ptr->xAxis.axisStart;

   slope_y = (ptr->yAxis.axisStart - ptr->yAxis.axisEnd)
                / (ptr->yAxis.max - ptr->yAxis.min);
   intercept_y = ptr->yAxis.axisEnd;

   for (i = 0 ; i < ptr->seriesCount ; i++)
      {
      hpen = CreatePen((series + i)->lineStyle, 1, (series + i)->lineColor);

      hprevpen = SelectObject(hdc, hpen);

      points = (POINTSTRUCT *) GlobalLock((series + i)->hPoints);
      pointCount = (series + i)->pointCount;

      if (pointCount > 0)
         {
         percent_x = slope_x * (points->x - ptr->xAxis.min)
                     + intercept_x;
         percent_y = slope_y * (points->y - ptr->yAxis.min)
                     + intercept_y;

         MoveToEx(hdc, PercentToPixelX(rect, percent_x),
                     PercentToPixelY(rect, percent_y), NULL);
         }

      for (j = 1 ; j < pointCount ; j++)
                      {
                      percent_x = slope_x * (points[j].x - ptr->xAxis.min)
                      + intercept_x;
         percent_y = slope_y * (points[j].y - ptr->yAxis.min)
                     + intercept_y;

         LineTo(hdc, PercentToPixelX(rect, percent_x),
                     PercentToPixelY(rect, percent_y));
         }

      GlobalUnlock((series + i)->hPoints);

      SelectObject(hdc, hprevpen);

      DeleteObject(hpen);
      }

   GlobalUnlock(ptr->hSeries);
   GlobalUnlock(hPtr);

   return TRUE;
}


#pragma argsused
BOOL CALLBACK AboutDlgProc(HWND hWndDlg, UINT msg, WPARAM wParam,
                           LPARAM lParam)
{
   switch(msg)
      {
```



```
      case WM_INITDIALOG:
         return TRUE;

      case WM_COMMAND:
         switch (wParam)
            {
            case IDC_OK:
               EndDialog(hWndDlg, 1);
               return TRUE;

            default:
               return FALSE;
            }

      default:
         return FALSE;
      }
}
```

```
// ========================================================================
//                                INDIVAT.C
// ========================================================================
// ========================================================================
//     Handles the display of the individual compound dialog box
// ------------------------------------------------------------------------
// ========================================================================
//                            Revision History
// ========================================================================
//  09/30/94    Radian(RLB)              Initial Revision
// ========================================================================

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

#include "landwin.h"
#include "str_ops.h"
#include "helpdef.h"

#define UP_3D    1
#define DOWN_3D  2


BOOL CALLBACK DialogProc(HWND hWndDlg, unsigned uiMsg, WORD wParam,
                         LONG lParam);


HGLOBAL  hATCopy;
AIRTOXIC *ATCopy;
int ATCopyCount;

int GetIndivATInfo(HWND hWnd, HINSTANCE hInst)
{
FARPROC DlgProcInstance;
int ret_value;
void CopyAT(void), UpdateAT(void), FreeAT(void);

   CopyAT();

   DlgProcInstance = MakeProcInstance(DialogProc, hInst);
   ret_value = DialogBox(hInst, "SELECTED_AT_DIALOG", hWnd, DlgProcInstance);
   FreeProcInstance(DlgProcInstance);

   if (ret_value) UpdateAT();
   else FreeAT();

   return ret_value;
}



void CopyAT(void)
{
HWND ActiveWindow;
HGLOBAL hOperData = NULL;
OPERDATASTRUCT *OperData;
AIRTOXIC *AirToxic;

   ATCopy = NULL;
   ActiveWindow = SendMessage(RetrieveClientWindow(), WM_MDIGETACTIVE,
                              0, 0);
   if (ActiveWindow != NULL)
      hOperData = SendMessage(ActiveWindow, AM_GETOPERHANDLE, 0, 0);
```



```
   if (hOperData != NULL)
      OperData = (OPERDATASTRUCT *) GlobalLock(hOperData);

   if (OperData != NULL)
      {
      if (OperData->AirToxics.AirToxicCount == 0)
         {
         GlobalUnlock(hOperData);
         return;
         }

      ATCopyCount = OperData->AirToxics.AirToxicCount;

      hATCopy = (HGLOBAL) GlobalAlloc(GHND, ATCopyCount * sizeof(AIRTOXIC));
      if (hATCopy == NULL)
         {
         GlobalUnlock(hOperData);
         return;
         }

      ATCopy = (AIRTOXIC *) GlobalLock(hATCopy);
      AirToxic = (AIRTOXIC *) GlobalLock(OperData->AirToxics.hAirToxic);

      memcpy(ATCopy, AirToxic, ATCopyCount * sizeof(AIRTOXIC));

      GlobalUnlock(OperData->AirToxics.hAirToxic);
      }
}



void UpdateAT(void)
{
HWND ActiveWindow;
HGLOBAL hOperData = NULL;
OPERDATASTRUCT *OperData;
AIRTOXIC *AirToxic;

   ActiveWindow = SendMessage(RetrieveClientWindow(), WM_MDIGETACTIVE,
                              0, 0);
   if (ActiveWindow != NULL)
      hOperData = SendMessage(ActiveWindow, AM_GETOPERHANDLE, 0, 0);

   if (hOperData != NULL)
      OperData = (OPERDATASTRUCT *) GlobalLock(hOperData);

   if (OperData != NULL)
      {
      if (OperData->AirToxics.AirToxicCount == 0)
         {
         GlobalUnlock(hOperData);
         return;
         }

      if (OperData->AirToxics.hAirToxic != NULL)
         GlobalFree(OperData->AirToxics.hAirToxic);

      OperData->AirToxics.AirToxicCount = ATCopyCount;
      OperData->AirToxics.hAirToxic = hATCopy;

      GlobalUnlock(hATCopy);
      }
}
```

```
void FreeAT(void)
{
   GlobalUnlock(hATCopy);
   GlobalFree(hATCopy);
}



BOOL AppendAT(HWND hWndDlg)
{
char string[30];
HGLOBAL hNewAT;
AIRTOXIC *NewAT;

   hNewAT = (HGLOBAL) GlobalAlloc(GHND, (ATCopyCount + 1) * sizeof(AIRTOXIC));
   if (hNewAT == NULL)
      {
      MessageBox(NULL, "Unable to create a new air toxic", "Memory Error",
                 MB_OK | MB_ICONSTOP);
      return FALSE;
      }

   NewAT = (AIRTOXIC *) GlobalLock(hNewAT);

   memcpy(NewAT, ATCopy, ATCopyCount * sizeof(AIRTOXIC));

   GetDlgItemText(hWndDlg, IDC_CHEMNAME, (NewAT + ATCopyCount)->Name,
                  sizeof((NewAT + ATCopyCount)->Name) - 1);
   if (! strlen(strip_spaces((NewAT + ATCopyCount)->Name)))
      {
      MessageBox(NULL, "Name must not be blank", "Validation Error",
                 MB_OK | MB_ICONSTOP);
      GlobalUnlock(hNewAT);
      GlobalFree(hNewAT);
      return FALSE;
      }

   GetDlgItemText(hWndDlg, IDC_MOLEWT, string, sizeof(string) - 1);
   (NewAT + ATCopyCount)->MoleWt = strtod(string, NULL);

   if ((NewAT + ATCopyCount)->MoleWt <= 0)
      {
      MessageBox(NULL, "Molecular Weight must be greater than zero", "Validation
 Error",
                 MB_OK | MB_ICONSTOP);
      GlobalUnlock(hNewAT);
      GlobalFree(hNewAT);
      return FALSE;
      }

   GetDlgItemText(hWndDlg, IDC_AT_CODIS, string, sizeof(string) - 1);
   (NewAT + ATCopyCount)->EmFact[0] = strtod(string, NULL);

   if ((NewAT + ATCopyCount)->EmFact[0] > 1E6)
      {
      MessageBox(NULL, "Emission factor must be less than 1,000,000", "Validatio
n Error",
                 MB_OK | MB_ICONSTOP);
      GlobalUnlock(hNewAT);
      GlobalFree(hNewAT);
      return FALSE;
      }

   GetDlgItemText(hWndDlg, IDC_AT_NOCO, string, sizeof(string) - 1);
   (NewAT + ATCopyCount)->EmFact[1] = strtod(string, NULL);
```



```
   if ((NewAT + ATCopyCount)->EmFact[1] > 1E6)
      {
      MessageBox(NULL, "Emission factor must be less than 1,000,000", "Validatio
n Error",
                 MB_OK | MB_ICONSTOP);
      GlobalUnlock(hNewAT);
      GlobalFree(hNewAT);
      return FALSE;
      }

   GlobalUnlock(hATCopy);
   GlobalFree(hATCopy);

   hATCopy = hNewAT;
   ATCopy = NewAT;
   ATCopyCount++;

   return TRUE;
}



void DeleteAT(int Record)
{
AIRTOXIC *AirToxic;
HGLOBAL hNewAT;
AIRTOXIC *NewAT;

   if (ATCopyCount == 0) return;
   if (ATCopyCount == 1)
      {
      ATCopyCount = 0;
      GlobalUnlock(hNewAT);
      GlobalFree(hNewAT);

      hNewAT = NULL;
      return;
      }

   hNewAT = (HGLOBAL) GlobalAlloc(GHND, (ATCopyCount - 1) * sizeof(AIRTOXIC));
   if (hNewAT == NULL)
      {
      MessageBox(NULL, "Unable to create a new air toxic", "Memory Error",
                 MB_OK | MB_ICONSTOP);
      return;
      }

   NewAT = (AIRTOXIC *) GlobalLock(hNewAT);

   if (Record > 0) memcpy(NewAT, ATCopy, Record * sizeof(AIRTOXIC));
   if (Record < (ATCopyCount - 1))
      memcpy(NewAT + Record, ATCopy + Record + 1,
             (ATCopyCount - Record - 1) * sizeof(AIRTOXIC));

   GlobalUnlock(hATCopy);
   GlobalFree(hATCopy);

   ATCopy = NewAT;
   hATCopy = hNewAT;
   ATCopyCount--;
}
```

```
#pragma argsused

static BOOL CALLBACK DialogProc(HWND hWndDlg, unsigned uiMsg, WORD wParam,
                                LONG lParam)
{
char string[30];
HWND ActiveWindow;
OPERDATASTRUCT *OperData;
static int CurrentAT;
int LoadATValues(HWND hWndDlg, int Index);
void DeleteAT(int Record);
BOOL AppendAT(HWND hWndDlg);
LRESULT CALLBACK ATFPNumberEditProc(HWND hWnd, UINT iMessage, WPARAM wParam,
                                    LPARAM lParam);
LRESULT CALLBACK ATSTRNumberEditProc(HWND hWnd, UINT iMessage, WPARAM wParam,
                                    LPARAM lParam);

   switch (uiMsg)
      {
      case WM_INITDIALOG:
         CenterWindow(hWndDlg);

         CustomizeControl(hWndDlg, IDC_CHEMNAME, (FARPROC) ATSTRNumberEditProc);
         CustomizeControl(hWndDlg, IDC_MOLEWT,   (FARPROC) ATFPNumberEditProc);
         CustomizeControl(hWndDlg, IDC_AT_CODIS, (FARPROC) ATFPNumberEditProc);
         CustomizeControl(hWndDlg, IDC_AT_NOCO,  (FARPROC) ATFPNumberEditProc);

         CurrentAT = LoadATValues(hWndDlg, 0);
         return TRUE;

      case WM_COMMAND:
         if (HIWORD(lParam) == EN_KILLFOCUS)
            {
            PostMessage(hWndDlg, AM_CONFIRMDATA, LOWORD(wParam), 0);
            return FALSE;
            }

         switch (LOWORD(wParam))
            {
            case IDC_OK:
               EndDialog(hWndDlg, TRUE);
               return TRUE;

            case IDC_CANCEL:
               EndDialog(hWndDlg, FALSE);
               return TRUE;

            case IDC_HELP:
               WinHelp(hWndDlg, "LANDWIN.HLP", HELP_CONTEXT, ATEditor);
               return TRUE;

            case IDC_ENDBTN:
            case IDC_STARTBTN:
            case IDC_REVERSEBTN:
            case IDC_FORWARDBTN:
               if (LOWORD(wParam) == IDC_STARTBTN) CurrentAT = 0;
               if (LOWORD(wParam) == IDC_ENDBTN)   CurrentAT = -2;
               if (LOWORD(wParam) == IDC_FORWARDBTN) ++CurrentAT;
               if (LOWORD(wParam) == IDC_REVERSEBTN) --CurrentAT;

               CurrentAT = LoadATValues(hWndDlg, CurrentAT);
               return TRUE;
```



```
            case IDC_APPENDBTN:
               AppendAT(hWndDlg);
               CurrentAT = LoadATValues(hWndDlg, ATCopyCount - 1);
               return TRUE;

            case IDC_DELETEBTN:
               DeleteAT(CurrentAT);
               CurrentAT = LoadATValues(hWndDlg, CurrentAT);
               return TRUE;

            default:
               break;
            }
      }

   return FALSE;
}


int LoadATValues(HWND hWndDlg, int Index)
{
char string[30];

   if (ATCopyCount == 0)
      {
      sprintf(string, "%4d",  0);
      SetDlgItemText(hWndDlg, IDC_RECORDNO,     "");
      SetDlgItemText(hWndDlg, IDC_RECORDCOUNT,  "");
      SetDlgItemText(hWndDlg, IDC_CHEMNAME,     "");
      SetDlgItemText(hWndDlg, IDC_MOLEWT,       "");
      SetDlgItemText(hWndDlg, IDC_AT_NOCO,      "");
      SetDlgItemText(hWndDlg, IDC_AT_CODIS,     "");
      return 0;
      }

   if (Index > (ATCopyCount - 1)) Index = ATCopyCount - 1;
   if ((Index < 0) && (Index != -2)) Index = 0;
   if (Index == -2) Index = ATCopyCount - 1;

   SetDlgItemText(hWndDlg, IDC_CHEMNAME, (ATCopy + Index)->Name);

   sprintf(string, "%.2f", (ATCopy + Index)->MoleWt);
   SetDlgItemText(hWndDlg, IDC_MOLEWT, string);

   sprintf(string, "%.2f", (ATCopy + Index)->EmFact[0]);
   SetDlgItemText(hWndDlg, IDC_AT_NOCO, string);

   sprintf(string, "%.2f", (ATCopy + Index)->EmFact[1]);
   SetDlgItemText(hWndDlg, IDC_AT_CODIS, string);

   sprintf(string, "%4d",  Index + 1);
   SetDlgItemText(hWndDlg, IDC_RECORDNO, string);

   sprintf(string, "%4d",  ATCopyCount);
   SetDlgItemText(hWndDlg, IDC_RECORDCOUNT, string);

   return Index;
}



static void ValidationError(char *string)
{
```

```
   MessageBox(NULL, string, "Validation Error", MB_ICONSTOP | MB_OK);
}



static LRESULT CALLBACK ATFFNumberEditProc(HWND hWnd, UINT iMessage, WPARAM wPar
am,
                                           LPARAM lParam)
{
HWND ActiveWindow;
char string[30];
LRESULT ret;
void UnCustomizeControl(HWND hWnd);
LRESULT CALLBACK CallSCProc(HWND hWnd, UINT iMessage, WORD wParam,
                            DWORD lParam);
NCCALCSIZE_PARAMS *lpncsp = (NCCALCSIZE_PARAMS *) lParam;
WINDOWPOS *WinPos;

   switch (iMessage)
      {
      case WM_GETDLGCODE:
         ret = CallSCProc(hWnd, iMessage, wParam, lParam);
         if (lParam)
            {
            if (((MSG *) lParam)->message == WM_KEYDOWN)
               if (((MSG *) lParam)->wParam == VK_RETURN) ret |= DLGC_WANTMESSAG
E;
            }
         return ret;

      case WM_NCDESTROY:
         UnCustomizeControl(hWnd);
         return TRUE;

      case WM_CHAR:
         memset(string, 0, sizeof(string));
         SendMessage(hWnd, WM_GETTEXT, sizeof(string), (long) string);

         *(string + strlen(string)) = (char) wParam;

         if (isprint((char) wParam))
            {
            if (is_string_pos_number(string))
                 return CallSCProc(hWnd, iMessage, wParam, lParam);
            else return TRUE;
            }

         return CallSCProc(hWnd, iMessage, wParam, lParam);

      case WM_NCPAINT:
         Draw3DFrame(hWnd, DOWN_3D);
         return TRUE;

      case WM_NCCALCSIZE:
         if (! (BOOL) wParam) return FALSE;

         WinPos = lpncsp->lppos;

         CopyRect(((RECT *) lpncsp->rgrc) + 2, lpncsp->rgrc);

         lpncsp->rgrc[1].left += 2;
         lpncsp->rgrc[1].right -= 2;
         lpncsp->rgrc[1].top += 2;
         lpncsp->rgrc[1].bottom -= 2;
```



```
         lpncsp->rgrc[2].left += 2;
         lpncsp->rgrc[2].right -= 2;
         lpncsp->rgrc[2].top += 2;
         lpncsp->rgrc[2].bottom -= 2;

         CopyRect(lpncsp->rgrc, ((RECT *) lpncsp->rgrc) + 2);

         lpncsp->rgrc[0].left = WinPos->x + 2;
         lpncsp->rgrc[0].top = WinPos->y + 2;
         lpncsp->rgrc[0].right = WinPos->cx - 2;
         lpncsp->rgrc[0].bottom = WinPos->cy - 2;

         return WVR_VALIDRECTS & WVR_HREDRAW & WVR_VREDRAW;

      case WM_PAINT:
         CallSCProc(hWnd, iMessage, wParam, lParam);
         Draw3DFrame(hWnd, DOWN_3D);
         return FALSE;

       case WM_KEYDOWN:
         switch (wParam)
            {
            case VK_RETURN:
               PostMessage(GetParent(hWnd), AM_CONFIRMDATA,
                                   GetDlgCtrlID(hWnd), 0);
               return TRUE;

            default:
               return CallSCProc(hWnd, iMessage, wParam, lParam);
            }

      default:
         return CallSCProc(hWnd, iMessage, wParam, lParam);
      }
}



static LRESULT CALLBACK ATSTRNumberEditProc(HWND hWnd, UINT iMessage, WPARAM wPa
ram,
                                           LPARAM lParam)
{
HWND ActiveWindow;
char string[30];
LRESULT ret;
void UnCustomizeControl(HWND hWnd);
LRESULT CALLBACK CallSCProc(HWND hWnd, UINT iMessage, WORD wParam,
                            DWORD lParam);
NCCALCSIZE_PARAMS *lpncsp = (NCCALCSIZE_PARAMS *) lParam;
WINDOWPOS *WinPos;

   switch (iMessage)
      {
      case WM_GETDLGCODE:
         ret = CallSCProc(hWnd, iMessage, wParam, lParam);
         if (lParam)
            {
            if (((MSG *) lParam)->message == WM_KEYDOWN)
               if (((MSG *) lParam)->wParam == VK_RETURN) ret |= DLGC_WANTMESSAG
E;
            }
         return ret;

      case WM_NCDESTROY:
         UnCustomizeControl(hWnd);
```

```
         return TRUE;

      case WM_NCPAINT:
         Draw3DFrame(hWnd, DOWN_3D);
         return TRUE;

      case WM_NCCALCSIZE:
         if (! (BOOL) wParam) return FALSE;

         WinPos = lpncsp->lppos;

         CopyRect(((RECT *) lpncsp->rgrc) + 2, lpncsp->rgrc);

         lpncsp->rgrc[1].left += 2;
         lpncsp->rgrc[1].right -= 2;
         lpncsp->rgrc[1].top += 2;
         lpncsp->rgrc[1].bottom -= 2;

         lpncsp->rgrc[2].left += 2;
         lpncsp->rgrc[2].right -= 2;
         lpncsp->rgrc[2].top += 2;
         lpncsp->rgrc[2].bottom -= 2;

         CopyRect(lpncsp->rgrc, ((RECT *) lpncsp->rgrc) + 2);

         lpncsp->rgrc[0].left = WinPos->x + 2;
         lpncsp->rgrc[0].top = WinPos->y + 2;
         lpncsp->rgrc[0].right = WinPos->cx - 2;
         lpncsp->rgrc[0].bottom = WinPos->cy - 2;

         return WVR_VALIDRECTS & WVR_HREDRAW & WVR_VREDRAW;

      case WM_PAINT:
         CallSCProc(hWnd, iMessage, wParam, lParam);
         Draw3DFrame(hWnd, DOWN_3D);
         return FALSE;

       case WM_KEYDOWN:
         switch (wParam)
              {
              case VK_RETURN:
                 PostMessage(GetParent(hWnd), AM_CONFIRMDATA,
                                   GetDlgCtrlID(hWnd), 0);
                 return TRUE;

              default:
                 return CallSCProc(hWnd, iMessage, wParam, lParam);
              }

      default:
         return CallSCProc(hWnd, iMessage, wParam, lParam);
      }
}
```

```
// ============================================================================
//                                KCALCFN.C
// ============================================================================
// ============================================================================
//                  Function for calculation the k value.
// ============================================================================
// ============================================================================
//                              Revision History
// ============================================================================
//  09/30/94    Radian(RLB)             Initial Revision
// ============================================================================

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

#include "windows.h"
#include "landwin.h"

/* ========================================================================== */
/*          Determine the calculation limitations of the system.              */
/* ========================================================================== */

#define FINE_ITERATIONS 10000
#define COARSE_ITERATIONS 50
#define TOLERANCE 0.000001

#define PI 3.1415265

/* ========================================================================== */
/*  This function bears a little description.  The function cannot be         */
/*  solved analytically for k.  Rather, for a given set of parameters,        */
/*  k must be approximated.                                                   */
/*                                                                            */
/*  The function can be shown to have two roots.  One root lies in the        */
/*  interval 0 and 1/refuse_age (the maximum of the function) and the         */
/*  other between 1/refuse_age and infinity.  The root of interest lies       */
/*  closest to the origin.  Therefore, in order to insure the correct         */
/*  root is found, this function implements two different approximation       */
/*  methods for the root:  BISECTION (used to adjust the approximation        */
/*  corsely) and NEWTON-RAPHSON (used to make fine adjustments to the         */
/*  approximation.                                                            */
/*                                                                            */
/*  BISECTION attempts to find the root on an interval by adding or           */
/*  subtracting ever decreasing corrections to a guess lying on the           */
/*  interval in an attempt to zero in on the root.                            */
/*                                                                            */
/*  After a rough but close estimation of the root, NEWTON-RAPHSON makes      */
/*  changes to the estimation be altering the value based on the              */
/*  derivative of the function.  Given a value close to the root, this        */
/*  method requires only a few iterations to achieve acceptable accuracy.    */
/* ========================================================================== */

double CalculateK(METHOD2E *Method2E, MODELPARAMETERS *Model)
{
double affected_depth, refuse_volume, affected_refuse_mass;
double adjusted_10, old_k, new_k, delta_k, correction;
double old_res, new_res;
double max_value;

int count = 0;

double landfill_function(double k, double refuse_age, double flow_rate,
                         double adjusted_10, double affected_refuse_mass);
double landfill_derivative(double k, double refuse_age);
```



```
/* ========================================================================== */
/*  The affected depth cannot exceed the landfill depth.  This assumes        */
/*  the well depth is 0.75 of the landfill depth in order to determine if     */
/*  this is so.                                                               */
/* ========================================================================== */

   affected_depth = Method2E->WellDepth + Method2E->ROI;

   if (affected_depth > Method2E->WellDepth / 0.75)
      affected_depth = Method2E->WellDepth / 0.75;

/* ========================================================================== */
/*            Calculate parameters required for the equation.                 */
/* ========================================================================== */

   refuse_volume = PI * affected_depth * Method2E->ROI * Method2E->ROI;

   affected_refuse_mass = refuse_volume * Method2E->Density;

   adjusted_10 = Method2E->Fraction * Model->10;

/* ========================================================================== */
/*  If the landfill function is less than zero at the local maximum           */
/*  (which can be shown to be 1/refuse_age), no solution can be found.        */
/*  Automatically return a -99999.                                            */
/* ========================================================================== */

  if ((max_value = landfill_function((1 / Method2E->RefuseAge),
                       Method2E->RefuseAge,
                       Method2E->FlowRate,
                       adjusted_10,
                       affected_refuse_mass)) < 0) return(-99999);

/* ========================================================================== */
/*  Just in case the parameters are already such that the maximum is the      */
/*  zero of the function, check to see and return it.                         */
/* ========================================================================== */


   if (max_value < TOLERANCE) return(max_value);

/* ========================================================================== */
/*  Set up for the BISECTION algorithm.  The initial correction value is      */
/*  set to 1/refuse_age (the maximum of the function).  Two values for k      */
/*  are estimated.  For the zero case, the landfill function will return      */
/*  its most negative value (on the interval of interest).  The direction     */
/*  of search is set initially to increasing k value.                         */
/* ========================================================================== */

   delta_k = 1 / (2 * Method2E->RefuseAge);

   old_k = 0.0;

   old_res = landfill_function(old_k,
                               Method2E->RefuseAge,
                               Method2E->FlowRate,
                               adjusted_10,
                               affected_refuse_mass);


   old_k = delta_k;

   while (count++ < COARSE_ITERATIONS)
      |
/* ========================================================================== */
/*      Determine the function value using the corrected value of k.         */
/* ========================================================================== */
```

```
      new_res = landfill_function(old_k,
                                  Method2E->RefuseAge,
                                  Method2E->FlowRate,
                                  adjusted_l0,
                                  affected_refuse_mass);

/* ================================================================== */
/*  If we manage to determine the k value accurately enough here, return  */
/*  the value and ignore the rest of the function.                    */
/* ================================================================== */

      if (fabs(new_res) < TOLERANCE) return(old_k);

/* ================================================================== */
/*  As soon as the new value of the landfill function changes sign from   */
/*  old value, reverse the correction direction and half the correction   */
/*  factor.                                                           */
/* ================================================================== */

      if (old_res * new_res < 0.0) delta_k = -(delta_k) / 2.0;

/* ================================================================== */
/*  Correct the old value of k and set the new result to the old result   */
/* ================================================================== */

      old_k += delta_k;
      old_res = new_res;
      }

/* ================================================================== */
/*  Here begins the NEWTON-RAPHSON step.  Because BISECTION is slow, we   */
/*  can't wait until a root is found with an acceptable tolerance.  Use   */
/*  this portion to refine the root.                                  */
/* ================================================================== */

   while (count++ < FINE_ITERATIONS)
      {
      correction = landfill_function(old_k,
                                     Method2E->RefuseAge,
                                     Method2E->FlowRate,
                                     adjusted_l0,
                                     affected_refuse_mass);

      correction /= landfill_derivative(old_k, Method2E->RefuseAge);

      new_k = old_k - correction;

      if ((old_k - new_k) < TOLERANCE) return(new_k);

      old_k = new_k;
      }

/* ================================================================== */
/*       Returns a -99999 if no value was found after both methods.       */
/* ================================================================== */

   return(-99999);
}


double landfill_function(k, refuse_age, flow_rate, adjusted_l0,
                         affected_refuse_mass)
double k, refuse_age, flow_rate, adjusted_l0, affected_refuse_mass;
{
double term1, term2;
```



```
   term1 = k * exp(-k * refuse_age);

   term2 = 525960.0 * (flow_rate / (2 * adjusted_l0 * affected_refuse_mass));

   return(term1 - term2);
}


double landfill_derivative(k, refuse_age)
double k, refuse_age;
{
double term1, term2;

   term1 = exp(-k * refuse_age);

   term2 = (1 - refuse_age * k);

   return(term1 * term2);
}
```

```
// ========================================================================
//                                KEDIT.C
// ========================================================================
// ========================================================================
//                    Allows the user to edit the k value.
// ========================================================================
// ========================================================================
//                              Revision History
// ========================================================================
//  09/30/94     Radian(RLB)              Initial Revision
// ========================================================================
#include <windows.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <ctype.h>

#include "landwin.h"
#include "str_ops.h"
#include "helpdef.h"


BOOL CALLBACK DialogProc(HWND hWndDlg, unsigned uiMsg, WORD wParam,
                         LONG lParam);


int GetKValue(HWND hWnd, HINSTANCE hInst)
{
FARPROC DlgProcInstance;
int ret_value;

   DlgProcInstance = MakeProcInstance(DialogProc, hInst);
   ret_value = DialogBox(hInst, "K_DIALOG", hWnd, DlgProcInstance);
   FreeProcInstance(DlgProcInstance);

   return ret_value;
}


#pragma argsused

static BOOL CALLBACK DialogProc(HWND hWndDlg, unsigned uiMsg, WORD wParam,
                                LONG lParam)
{
HWND ActiveWindow;
HGLOBAL hOperData = NULL;
OPERDATASTRUCT *OperData;
char string[30];
HINSTANCE hInst;

   switch (uiMsg)
      {
      case WM_INITDIALOG:
         CenterWindow(hWndDlg);

         ActiveWindow = SendMessage(RetrieveClientWindow(), WM_MDIGETACTIVE,
                                    0, 0);
         if (ActiveWindow != NULL)
            hOperData = SendMessage(ActiveWindow, AM_GETOPERHANDLE, 0, 0);

         if (hOperData != NULL)
            OperData = (OPERDATASTRUCT *) GlobalLock(hOperData);


         sprintf(string, "%.4g", OperData->Model.k);
         SetDlgItemText(hWndDlg, IDC_VALUEEDIT, (LPCSTR) string);
```



```
         CustomizeControl(hWndDlg, IDC_VALUEEDIT, (FARPROC) FPNumberEditProc);

         if (hOperData != NULL) GlobalUnlock(hOperData);
         return TRUE;

      case WM_COMMAND:
         switch (LOWORD(wParam))
            {
            case IDC_OK:
               GetDlgItemText(hWndDlg, IDC_VALUEEDIT, (LPSTR) string,
                              sizeof(string));
               ActiveWindow = SendMessage(RetrieveClientWindow(),
                                          WM_MDIGETACTIVE, 0, 0);
               if (ActiveWindow != NULL)
                  hOperData = SendMessage(ActiveWindow, AM_GETOPERHANDLE, 0, 0);

               if (hOperData != NULL)
                  OperData = (OPERDATASTRUCT *) GlobalLock(hOperData);

               OperData->Model.k = strtod(string, NULL);

               if (hOperData != NULL) GlobalUnlock(hOperData);

               EndDialog(hWndDlg, TRUE);
               return TRUE;

            case IDC_CANCEL:
               EndDialog(hWndDlg, FALSE);
               return TRUE;

            case IDC_HELP:
               WinHelp(hWndDlg, "LANDWIN.HLP", HELP_CONTEXT, ParametersUser);
               return TRUE;

            case IDC_VALUEEDIT:
               return FALSE;

            default:
               break;
            }
      }

   return FALSE;
}
```

```
// ========================================================================
//                                LANDCUST.C
// ========================================================================
// ========================================================================
//  Creates the container custom control for reserving toolbar space and
//  placeing titles on the operational parameters screen.
// ========================================================================
// ========================================================================
//                            Revision History
// ========================================================================
//  09/30/94     Radian(RLB)              Initial Revision
// ========================================================================

#include <windows.h>


#include "landwin.h"

#define UP_3D    1
#define DOWN_3D  2

void Draw3DFrame(HWND hWnd, int iStyle);



long FAR PASCAL ContainerWndProc(HWND hWnd, WORD iMessage, WPARAM wParam,
                                 LPARAM lParam);



void RegisterLandwinCustom(void)
{
void RegisterContainer(void);

   RegisterCont[illegible]er();
}


void UnregisterLandwinCustom(void)
{
   UnregisterClass("LANDCNT", RetrieveAppInstance());
}



void RegisterContainer(void)
{
WNDCLASS WndClass;

   WndClass.cbClsExtra = 0;
   WndClass.cbWndExtra = sizeof(long);
   WndClass.hbrBackground = GetStockObject(LTGRAY_BRUSH);
   WndClass.hCursor = LoadCursor(NULL, IDC_ARROW);
   WndClass.hIcon = NULL;
   WndClass.hInstance = RetrieveAppInstance();
   WndClass.lpfnWndProc = (WNDPROC) ContainerWndProc;
   WndClass.lpszClassName = "LANDCNT";
   WndClass.lpszMenuName = NULL;
   WndClass.style = CS_HREDRAW | CS_VREDRAW;

   RegisterClass(&WndClass);
}
```



```
long FAR PASCAL ContainerWndProc(HWND hWnd, WORD iMessage, WPARAM wParam,
                                 LPARAM lParam)
{
HDC hDC;
HBRUSH hOld;
HRGN ClipRgn;
HWND hWndParent;
RECT ParentRect, ThisRect, ClientRect, SaveRect;
int Width, Height;
long style;
NCCALCSIZE_PARAMS *lpncsp = (NCCALCSIZE_PARAMS *) lParam;
WINDOWPOS *WinPos;

   switch (iMessage)
      {
      case WM_CREATE:
         return 0;

      case WM_SETFOCUS:
         SetFocus(GetWindow(hWnd, GW_CHILD));
         return TRUE;

      case WM_NCCALCSIZE:
         if (! (BOOL) wParam) return FALSE;

         WinPos = lpncsp->lppos;

         CopyRect(((RECT *) lpncsp->rgrc) + 2, lpncsp->rgrc);

         lpncsp->rgrc[1].left += 2;
         lpncsp->rgrc[1].right -= 2;
         lpncsp->rgrc[1].top += 2;
         lpncsp->rgrc[1].bottom -= 2;

         lpncsp->rgrc[2].left += 2;
         lpncsp->rgrc[2].right -= 2;
         lpncsp->rgrc[2].top += 2;
         lpncsp->rgrc[2].bottom -= 2;

         CopyRect(lpncsp->rgrc, ((RECT *) lpncsp->rgrc) + 2);

         lpncsp->rgrc[0].left = WinPos->x + 2;
         lpncsp->rgrc[0].top = WinPos->y + 2;
         lpncsp->rgrc[0].right = WinPos->cx - 2;
         lpncsp->rgrc[0].bottom = WinPos->cy - 2;

         return WVR_VALIDRECTS & WVR_HREDRAW & WVR_VREDRAW;
/*
      case WM_ERASEBKGND:
         hWndParent = GetParent(hWnd);
         GetClientRect(hWndParent, &ParentRect);
         GetWindowRect(hWnd, &ThisRect);
         GetClientRect(hWnd, &ClientRect);

         Width = ThisRect.right - ThisRect.left;
         Height = ThisRect.bottom - ThisRect.top;

         style = GetWindowLong(hWnd, GWL_STYLE);
         if (style & CNT_SIZEX)
            Width = ParentRect.right - ParentRect.left
                    - ClientRect.left;
         if (style & CNT_SIZEY)
            Height = ParentRect.bottom - ParentRect.top
                     - ClientRect.top;
```

```
         hDC = GetDC(hWnd);
         hOld = SelectObject(hDC, GetStockObject(LTGRAY_BRUSH));
         Rectangle(hDC, ClientRect.left, ClientRect.top,
                        ClientRect.left + Width - 1, ClientRect.top + Height - 1
);
         SelectObject(hDC, hOld);
         ReleaseDC(hWnd, hDC);
         return FALSE;
*/
      case WM_NCPAINT:
         hWndParent = GetParent(hWnd);
         GetClientRect(hWndParent, &ParentRect);
         GetWindowRect(hWnd, &ThisRect);
         GetClientRect(hWnd, &ClientRect);

         Width = ThisRect.right - ThisRect.left;
         Height = ThisRect.bottom - ThisRect.top;

         style = GetWindowLong(hWnd, GWL_STYLE);
         if (style & CNT_SIZEX)
            Width = ParentRect.right - ParentRect.left
                        - ClientRect.left;
         if (style & CNT_SIZEY)
            Height = ParentRect.bottom - ParentRect.top
                        - ClientRect.top;

         SetWindowPos(hWnd, NULL, 0, 0,
                     Width, Height,
                     SWP_NOMOVE | SWP_NOZORDER | SWP_SHOWWINDOW | SWP_NOACTIVAT
E);

         Draw3DFrame(hWnd, UP_3D);
         return FALSE;
      }

   return DefWindowProc(hWnd, iMessage, wParam, lParam);
}
```

```
/* ========================================================================== */
/*
    Landwin.c
    ---------

    Description:

    This file contains the application initialization code, the main
    window handler, and the menu facility.  It also contains the code
    for some general purpose applications functions.

    Revision history:

    rlb    6/15/96      Initial revision for release

*/
/* ========================================================================== */


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

#include "landwin.h"
#include "landoper.h"

/* Global Data Definitions */

char StudyName[MAXPATH];
MODELLIMITS Limits;

/* Static Data */

static HINSTANCE hInst;  // Use RetrieveAppInstance() to obtain.
static HWND hWndFrame;       // UseRetrieveFrameWindow() to obtain.
static HWND hWndMDI;       // UseRetrieveClientWindow() to obtain.
static HWND hMenuNormal;
static HWND ActiveTool = NULL;   // Used to hold the active tool so we can
                                 // do the messages for it.
static HMENU Items[5];
static BOOL MenuExpanded = FALSE;

/* Generic Prototypes */


long FAR PASCAL WndProc(HWND hWnd, WORD iMessage, WPARAM wParam, LPARAM lParam);
extern long FAR PASCAL OperWndProc(HWND hWnd, WORD iMessage, WPARAM wParam,
                                   LPARAM lParam);
extern long FAR PASCAL GraphWndProc(HWND hWnd, WORD iMessage, WPARAM wParam,
                                   LPARAM lParam);
extern long FAR PASCAL TextWndProc(HWND hWnd, WORD iMessage, WPARAM wParam,
                                   LPARAM lParam);

void DefaultsToMenuSelections(HWND hWnd);
void ChoicesToMenuSelections(HWND hWnd);
void CheckLandwinDefaultMenu(HMENU hMenu, WORD id);
void CheckLandwinLMenu(HMENU hMenu, WORD id);
void EnableLandwinLMenu(HMENU hMenu, WORD id);
void CheckLandwinKMenu(HMENU hMenu, WORD id);
void EnableLandwinKMenu(HMENU hMenu, WORD id);
void SetDefaultKChoice(int Choice);
void SetDefaultLChoice(int Choice);
void SetDefaultNMOCChoice(int Choice);
void SetDefaultMode(int Choice);
```



```
void SetLandfillType(int Choice);
BOOL HandleMenuCommand(HWND hWnd, WORD Control);

/* ---------------------------------------------------------------------- */
/*

   Procedure:  WinMain

   Purpose:    Application Entry Point.  Creates MDI classes and
               initializes system.

*/
/* ---------------------------------------------------------------------- */


#pragma argsused
int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpszCmdParam,
                   int nCmdShow)
{
RECT r;
HDC hdc;
HACCEL hAccel;
MSG Message;
WNDCLASS WndClass;
WORD DlgWidth, DlgHeight;

/* Global to hold the instance of this invocation. */

   hInst = hInstance;

/* Register the MDI window class */

   if (! hPrevInstance)
      {
      WndClass.cbClsExtra = 0;
      WndClass.cbWndExtra = 0;
      WndClass.hbrBackground = GetStockObject(WHITE_BRUSH);
      WndClass.hCursor = LoadCursor(NULL, IDC_ARROW);
      WndClass.hIcon = LoadIcon(hInstance, "ICON_1");
      WndClass.hInstance = hInstance;
      WndClass.lpfnWndProc = (WNDPROC) WndProc;
      WndClass.lpszClassName = "LANDWIN";
      WndClass.lpszMenuName = "LANDWIN_OPER";
      WndClass.style = CS_HREDRAW | CS_VREDRAW;

      RegisterClass(&WndClass);

      WndClass.cbClsExtra = 0;
      WndClass.cbWndExtra = 3 * sizeof(long);
      WndClass.hbrBackground = GetStockObject(WHITE_BRUSH);
      WndClass.hCursor = LoadCursor(NULL, IDC_ARROW);
      WndClass.hIcon = LoadIcon(hInstance, "LANDOP_ICON");
      WndClass.hInstance = hInstance;
      WndClass.lpfnWndProc = (WNDPROC) OperWndProc;
      WndClass.lpszClassName = "OPERWINDOW";
      WndClass.lpszMenuName = NULL;
      WndClass.style = CS_HREDRAW | CS_VREDRAW;

      RegisterClass(&WndClass);


               WndClass.cbClsExtra = 0;
               WndClass.cbWndExtra = 3 * sizeof(long);
               WndClass.hbrBackground = GetStockObject(WHITE_BRUSH);
               WndClass.hCursor = LoadCursor(NULL, IDC_ARROW);
               WndClass.hIcon = LoadIcon(hInstance, "LANDGRAPH_ICON");
```

```
                WndClass.hInstance = hInstance;
                WndClass.lpfnWndProc = (WNDPROC) GraphWndProc;
                WndClass.lpszClassName = "GRAPHWINDOW";
                WndClass.lpszMenuName = NULL;
                WndClass.style = CS_HREDRAW | CS_VREDRAW;

                RegisterClass(&WndClass);

                WndClass.cbClsExtra = 0;
                WndClass.cbWndExtra = 3 * sizeof(long);
                WndClass.hbrBackground = GetStockObject(WHITE_BRUSH);
                WndClass.hCursor = LoadCursor(NULL, IDC_ARROW);
                WndClass.hIcon = LoadIcon(hInstance, "LANDTEXT_ICON");
                WndClass.hInstance = hInstance;
                WndClass.lpfnWndProc = (WNDPROC) TextWndProc;
                WndClass.lpszClassName = "TEXTWINDOW";
                WndClass.lpszMenuName = NULL;
                WndClass.style = CS_HREDRAW | CS_VREDRAW | CS_CLASSDC;

                RegisterClass(&WndClass);

        RegisterLandwinCustom();
        RegisterOperClass();
        }


/* Create the application main window */

   hWndFrame = CreateWindow("LANDWIN", "Landfill Air Emissions Estimation Model"
,
                        WS_OVERLAPPEDWINDOW | WS_CLIPCHILDREN,
                        CW_USEDEFAULT,                          /* X */
                        CW_USEDEFAULT,                          /* Y */
                        CW_USEDEFAULT,
                        CW_USEDEFAULT,
                        NULL,
                        NULL,
                        hInstance,
                        NULL);

/* Load the Default Parameter Data and save it out to disk in order */
/* to create the ini default values. */

   LoadModelLimits();
   SaveModelLimits();

/* Show the window */

   ShowWindow(hWndFrame, nCmdShow);
   UpdateWindow(hWndFrame);

/* Load the application accelerators so the user can use shortcut keys. */

   hAccel = LoadAccelerators(hInst, "ACCEL");

/* Application message loop.  This loop handles routing messages */
/* that are accelerators and will also route to the toolbar. */


   while (GetMessage(&Message, NULL, 0, 0))
      {
      if (! TranslateMDISysAccel(hWndMDI, &Message))
         {
         if ((ActiveTool != NULL))
            {
            if (!IsDialogMessage(GetWindow(ActiveTool, GW_CHILD), &Message))
```



```
               {
               TranslateMessage(&Message);
               DispatchMessage(&Message);
               }
            }
         else
            {
            if (! TranslateAccelerator(hWndFrame, hAccel, &Message))
               TranslateMessage(&Message);

            DispatchMessage(&Message);
            }
         }
      }

/* Unregister the MDI classes and custom controls */

   UnregisterLandwinCustom();
   UnregisterClass("OPERWINDOW", hInstance);
   UnregisterClass("GRAPHWINDOW", hInstance);
   UnregisterClass("LANDWIN", hInstance);

/* Return with last message value */

   return Message.wParam;
}


/* ---------------------------------------------------------------- */
/*

   Procedure:  WndProc

   Purpose:    Application Message Handler

*/
/* ---------------------------------------------------------------- */

long FAR PASCAL WndProc(HWND hWnd, WORD iMessage, WPARAM wParam, LPARAM lParam)
{
CLIENTCREATESTRUCT CS;
HMENU hMenu;
static HWND OperWnd, Reserve;
HWND ActiveWindow;
int WindowWidth, WindowHeight;
RECT r;
void DestroyTool(HWND ToolWnd);
HWND CreateTool(HWND Parent, LPCSTR DialogString, FARPROC DlgProc);
extern BOOL CALLBACK OperDlgProc(HWND hWndDlg, unsigned uiMsg, WORD wParam,
                                 LONG lParam);

     switch (iMessage)
        {
        case WM_CREATE:

/* Upon window creation, create the toolbar area, the client area, and */
/* remove the menu selections that aren't relevant until a study is */
/* available. */

           OperWnd = CreateTool(hWnd, (LPCSTR) "OPERTOOLDLG", (FARPROC) OperDlgP
roc);

           hMenuNormal = GetMenu(hWnd);

           CS.hWindowMenu = GetSubMenu(hMenuNormal, 6);
           CS.idFirstChild = 1200;
```

```
            hWndMDI = CreateWindow("MDICLIENT", "",
                        WS_CHILD | WS_CLIPCHILDREN | WS_VISIBLE |
                        WS_VSCROLL | WS_HSCROLL | MDIS_ALLCHILDSTYLES,
                        0, 0, 0, 0, hWnd, NULL,
                        hInst, &CS);

            Items[0] = GetSubMenu(hMenuNormal, 1);
            Items[1] = GetSubMenu(hMenuNormal, 2);
            Items[2] = GetSubMenu(hMenuNormal, 3);
            Items[3] = GetSubMenu(hMenuNormal, 4);
            Items[4] = GetSubMenu(hMenuNormal, 6);

            RemoveMenu(hMenuNormal, 1, MF_BYPOSITION);
            RemoveMenu(hMenuNormal, 1, MF_BYPOSITION);
            RemoveMenu(hMenuNormal, 1, MF_BYPOSITION);
            RemoveMenu(hMenuNormal, 1, MF_BYPOSITION);
            RemoveMenu(hMenuNormal, 2, MF_BYPOSITION);

/* Create the toolbar container window */

            GetWindowRect(OperWnd, &r);

            Reserve = CreateWindow("LANDCNT", "",
                                   WS_CHILD | WS_CLIPCHILDREN | CNT_SIZEX | WS_V
ISIBLE,
                                   0, 0,
                                   r.right - r.left,
                                   r.bottom - r.top,
                                   hWnd, NULL, hInst, NULL);

            SetParent(OperWnd, Reserve);
            return FALSE;

         case WM_CLOSE:
            DestroyWindow(hWnd);
            return FALSE;

         case WM_DESTROY:
            DestroyTool(OperWnd);
            PostQuitMessage(0);
            return FALSE;

         case WM_SIZE:
            if (IsWindow(hWndMDI) != NULL)
               ActiveWindow = SendMessage(hWndMDI, WM_MDIGETACTIVE, 0, 0L);
            else ActiveWindow = NULL;

/* Make sure the client area window is in the space not occupied by */
/* the tool bar. */

            if (IsWindow(hWndMDI))
               {
               WindowWidth = LOWORD(lParam);
               WindowHeight = HIWORD(lParam);

               GetWindowRect(Reserve, &r);

               SetWindowPos(hWndMDI, NULL, 0, r.bottom - r.top, WindowWidth,
                           WindowHeight - r.bottom + r.top,
                           SWP_NOZORDER | SWP_SHOWWINDOW);
               }

/* Turn on (or off) the toolbar associated with the active window type */

            if (ActiveWindow != NULL)
```



```
               {
               switch (WhichMDIType(ActiveWindow))
                  {
                  case OPERWINDOW:
                     if ((! IsWindowVisible(OperWnd)) && (ActiveTool != NULL))
                        {
                        ShowWindow(ActiveTool, SW_HIDE);
                        ActiveTool = NULL;
                        }

                     if (! IsWindowVisible(OperWnd))
                        {
                        ActiveTool = OperWnd;
                        ShowWindow(OperWnd, SW_NORMAL);
                        InvalidateRect(OperWnd, NULL, 0);
                        }
                     else PostMessage(OperWnd, AM_VALUESCHANGED, 0, 0);

                     break;

                  case NULL:
                  default:
                     if (ActiveTool != NULL) ShowWindow(ActiveTool, SW_HIDE);

                     WindowWidth = LOWORD(lParam);
                     WindowHeight = HIWORD(lParam);

                     GetWindowRect(OperWnd, &r);

//                     SetWindowPos(hWndMDI, NULL, 0, 0, WindowWidth,
//                                  WindowHeight,
//                                  SWP_NOZORDER | SWP_SHOWWINDOW);

                     ActiveTool = NULL;

                     break;
                  }
               }
            else
               {
               if (ActiveTool != NULL) ShowWindow(ActiveTool, SW_HIDE);

               WindowWidth = LOWORD(lParam);
               WindowHeight = HIWORD(lParam);

//               if (IsWindow(hWndMDI))
//                  SetWindowPos(hWndMDI, NULL, 0, 0, WindowWidth,
//                               WindowHeight,
//                               SWP_NOZORDER | SWP_SHOWWINDOW);

               ActiveTool = NULL;
               }

            return TRUE;

         case AM_WINDOWACTIVE:
/* The WM_SIZE message handles the display and hiding of the toolbar */
/* If the MDI client window is active, we need to dummy up a WM_SIZE */
/* message to have the system display the toolbar. */

/* Resize the MDI Window */

            GetClientRect(hWnd, &r);
            lParam = MAKELPARAM(r.right - r.left, r.bottom - r.top);
            SendMessage(hWnd, WM_SIZE, 0, lParam);
```

```
/* If a study is being looked at (ie., window not NULL), make sure the */
/* menu has all the choices they need.  If not, remove them */

          if (IsWindow(hWndMDI) != NULL)
             ActiveWindow = SendMessage(hWndMDI, WM_MDIGETACTIVE, 0, 0L);
          else ActiveWindow = NULL;

          if (ActiveWindow != NULL)
             {
             ChoicesToMenuSelections(hWnd);

             if (! MenuExpanded)
                {
                hMenuNormal = GetMenu(hWnd);

                InsertMenu(hMenuNormal, 1, MF_BYPOSITION | MF_POPUP, Items[0],
                           "&Edit");
                InsertMenu(hMenuNormal, 2, MF_BYPOSITION | MF_POPUP, Items[1],
                           "&Defaults");
                InsertMenu(hMenuNormal, 3, MF_BYPOSITION | MF_POPUP, Items[2],
                           "&Parameters");
                InsertMenu(hMenuNormal, 4, MF_BYPOSITION | MF_POPUP, Items[3],
                           "&Reports");
                InsertMenu(hMenuNormal, 6, MF_BYPOSITION | MF_POPUP, Items[4],
                           "&Window");

                EnableMenuItem(hMenuNormal, LAND_MENU_SAVE, MF_ENABLED);
                EnableMenuItem(hMenuNormal, LAND_MENU_SAVEAS, MF_ENABLED);

                SetMenu(hWnd, hMenuNormal);
                DrawMenuBar(hWnd);

                MenuExpanded = TRUE;
                }
             }
          else
             {
             if (MenuExpanded)
                {
                RemoveMenu(hMenuNormal, 1, MF_BYPOSITION);
                RemoveMenu(hMenuNormal, 1, MF_BYPOSITION);
                RemoveMenu(hMenuNormal, 1, MF_BYPOSITION);
                RemoveMenu(hMenuNormal, 1, MF_BYPOSITION);
                RemoveMenu(hMenuNormal, 2, MF_BYPOSITION);


                EnableMenuItem(hMenuNormal, LAND_MENU_SAVE, MF_GRAYED);
                EnableMenuItem(hMenuNormal, LAND_MENU_SAVEAS, MF_GRAYED);

                DrawMenuBar(hWnd);
                MenuExpanded = FALSE;
                }
             }

          return TRUE;

       case AM_GETACTIVETOOL:
          return ActiveTool;

                case WM_COMMAND:
/* Windows MDI interface stuff */

          if (wParam >= 1200)
             {
             ActiveWindow = GetDlgItem(RetrieveClientWindow(), wParam);
             SendMessage(RetrieveClientWindow(), WM_MDIACTIVATE,
```



```
                         (WPARAM) ActiveWindow, 0);
             return(DefFrameProc(hWnd, hWndMDI, iMessage, wParam, lParam));
             }

          return HandleMenuCommand(hWnd, LOWORD(wParam));
          }

       return(DefFrameProc(hWnd, hWndMDI, iMessage, wParam, lParam));
}



/* ---------------------------------------------------------------- */
/*

   Procedure:  GetActiveMDIType

   Purpose:    Function to obtain the type of the currently active
               window in the MDI frame.

*/
/* ---------------------------------------------------------------- */


int GetActiveMDIType(void)
{
HWND ActiveWindow;

   if (IsWindow(hWndMDI) != NULL)
      ActiveWindow = SendMessage(hWndMDI, WM_MDIGETACTIVE, 0, 0L);
   else return NULLWINDOW;

   return WhichMDIType(ActiveWindow);
}



/* ---------------------------------------------------------------- */
/*

   Procedure:  WhichMDIType

   Purpose:    Determines the class of a window.

*/
/* ---------------------------------------------------------------- */


int WhichMDIType(HWND hWnd)
{
char Name[30];

   GetClassName(hWnd, Name, sizeof(Name));

   if (! strcmp(Name, "OPERWINDOW")) return OPERWINDOW;
   if (! strcmp(Name, "GRAPHWINDOW")) return GRAPHWINDOW;
   if (! strcmp(Name, "TEXTWINDOW")) return TEXTWINDOW;

   return NULLWINDOW;
}



/* ---------------------------------------------------------------- */
/*
```

```
   Procedure:  CreateTool

   Purpose:    Creates the toolbar area and tool windows that will
               occupy the area.

*/
/* ------------------------------------------------------------------ */


HWND CreateTool(HWND Parent, LPCSTR DialogString, FARPROC DlgProc)
{
HWND ToolWnd, ToolDlg;
DLGPROC ToolProc;
int DialogWidth, DialogHeight;
RECT r;

/* Create the container */

   ToolWnd = CreateWindow("LANDCNT", "",
                WS_CHILD | WS_CLIPCHILDREN | CNT_SIZEX,
                0, 0, 0, 0, Parent, NULL, hInst, NULL);

/* Create instance of the dialog */

   ToolProc = (DLGPROC) MakeProcInstance(DlgProc, hInst);
   ToolDlg = CreateDialog(hInst, DialogString, ToolWnd, ToolProc);

   SetWindowLong(ToolWnd, 0, (long) ToolProc);

   GetWindowRect(ToolDlg, &r);
   DialogWidth = r.right - r.left - 1;
   DialogHeight = r.bottom - r.top - 1;

   SetWindowPos(ToolWnd, NULL, 0, 0, DialogWidth, DialogHeight, SWP_NOZORDER);

   return ToolWnd;
}


/* ------------------------------------------------------------------ */
/*

   Procedure:  DestroyTool

   Purpose:    Destroys the toolbar

*/
/* ------------------------------------------------------------------ */



void DestroyTool(HWND ToolWnd)
{
DLGPROC DlgProc;

   DlgProc = (FARPROC) GetWindowLong(ToolWnd, 0);
   DestroyWindow(ToolWnd);
   FreeProcInstance(DlgProc);
}


/* ------------------------------------------------------------------ */
/*

   Procedure:  RetrieveAppInstance
```



```
   Purpose:    Returns the application instance to the caller.

*/
/* ------------------------------------------------------------------ */


HINSTANCE RetrieveAppInstance(void)
{
   return hInst;
}


/* ------------------------------------------------------------------ */
/*

   Procedure:  RetrieveClientWindow

   Purpose:    Returns the MDI client area window

*/
/* ------------------------------------------------------------------ */


HWND RetrieveClientWindow(void)
{
   return hWndMDI;
}


/* ------------------------------------------------------------------ */
/*

   Procedure:  RetrieveFrameWindow

   Purpose:    Returns the MDI frame window.

*/
/* ------------------------------------------------------------------ */



HWND RetrieveFrameWindow(void)
{
   return hWndFrame;
}



/* ------------------------------------------------------------------ */
/*

   Procedure:  ClearMessages

   Purpose:    Clears the application message queue.

*/
/* ------------------------------------------------------------------ */



void ClearMessages(void)
{
MSG Message;

   while (PeekMessage(&Message, NULL, 0, 0, PM_REMOVE))
      {
```

```
      TranslateMessage(&Message);
      DispatchMessage(&Message);
      }
}


/* ---------------------------------------------------------------- */
/*

   Procedure:   HandleMenuCommand

   Purpose:     Extension of the application message handler that
                deals with menu selections.

*/
/* ---------------------------------------------------------------- */



BOOL HandleMenuCommand(HWND hWnd, WORD Control)
{
HWND ActiveWindow;
HMENU hMenu;
int ret;
char Name[30];
int GetActiveMDIType(void);
void DisplayAbout(HWND hWnd, HINSTANCE hInst);
void DisplayCurrentParameters(HWND hWnd, HINSTANCE hInst);
BOOL GetKValue(HWND hWnd, HINSTANCE hInst);
BOOL GetLValue(HWND hWnd, HINSTANCE hInst);
BOOL GetOptions(HWND hWnd, HINSTANCE hInst);
HGLOBAL CalculationModule(int ChemIndex);
void RefuseEstimator(HWND hWnd, HINSTANCE hInst);
void CreateGraphic(int);
void CreateReport(int);

   hMenu = GetMenu(hWnd);

   switch (Control)
      {
      case LAND_MENU_OPEN:
         GetOldStudy();
         ChoicesToMenuSelections(hWnd);
         return TRUE;

      case LAND_MENU_NEW:
         GetNewStudy();
         ChoicesToMenuSelections(hWnd);
         return TRUE;

      case LAND_MENU_SAVE:
         switch (GetActiveMDIType())
            {
            case TEXTWINDOW:
            case GRAPHWINDOW:
               SendMessage(MDIGETACTIVE(), AM_SAVE, 0, 0);
               return TRUE;

            default:
               SaveStudy();
               return TRUE;
            }

      case LAND_MENU_SAVEAS:
         switch (GetActiveMDIType())
            {
```



```
            case TEXTWINDOW:
            case GRAPHWINDOW:
               SendMessage(MDIGETACTIVE(), AM_SAVE, 0, 0);
               return TRUE;

            default:
               SaveAsStudy();
               return TRUE;
            }

      case LAND_MENU_PRINT:
         switch (GetActiveMDIType())
            {
            case TEXTWINDOW:
            case GRAPHWINDOW:
               SendMessage(MDIGETACTIVE(), AM_PRINT, 0, 0);
               return TRUE;

            default:
               return FALSE;
            }

      case LAND_MENU_EXIT:
         PostMessage(hWnd, WM_CLOSE, 0, 0);
         return TRUE;

      case LAND_MENU_PASTE:
         if (GetActiveMDIType() != NULLWINDOW)
            {
            SendMessage(MDIGETACTIVE(), AM_PASTE, 0, 0);
            return TRUE;
            }
         else return FALSE;



      case LAND_MENU_CUT:
         if (GetActiveMDIType() != NULLWINDOW)
            {
            SendMessage(MDIGETACTIVE(), AM_CUT, 0, 0);
            return TRUE;
            }
         else return FALSE;


      case LAND_MENU_COPY:
         if (GetActiveMDIType() != NULLWINDOW)
            {
            SendMessage(MDIGETACTIVE(), AM_COPY, 0, 0);
            return TRUE;
            }
         else return FALSE;

      case LAND_MENU_REGULATORY:
      case LAND_MENU_AP42:
      case LAND_MENU_SPECIFIC:
         if (Control == LAND_MENU_REGULATORY) SetDefaultMode(REGULATORY);
         if (Control == LAND_MENU_AP42) SetDefaultMode(AP42);
         if (Control == LAND_MENU_SPECIFIC) SetDefaultMode(USER);

         CheckLandwinDefaultMenu(hMenu, Control);
         DefaultsToMenuSelections(hWnd);
         SetModelDefaults(MDIGETACTIVE());
         return TRUE;

      case LAND_MENU_CURRENT:
```

```
         DisplayCurrentParameters(hWnd, hInst);
         return TRUE;

      case LAND_MENU_DEFAULT_REG_K:
      case LAND_MENU_DEFAULT_AP42_K:
      case LAND_MENU_USER_K:
      case LAND_MENU_CALC_K:
         if (Control == LAND_MENU_DEFAULT_REG_K) SetDefaultKChoice(REGULATORY);
         if (Control == LAND_MENU_DEFAULT_AP42_K) SetDefaultKChoice(AP42);
         if (Control == LAND_MENU_USER_K)
            {
            GetKValue(hWnd, hInst);
            SetDefaultKChoice(USER);
            }
         if (Control == LAND_MENU_CALC_K)
            {
            CalculateKValue(hWnd, hInst);
            SetDefaultKChoice(CALCULATED);
            }

         CheckLandwinKMenu(hMenu, Control);
         return TRUE;

      case LAND_MENU_USER_L:
      case LAND_MENU_DEFAULT_REG_L:
      case LAND_MENU_DEFAULT_AP42_L:
         if (Control == LAND_MENU_USER_L)
            {
            GetLValue(hWnd, hInst);
            SetDefaultLChoice(USER);
            }
         if (Control == LAND_MENU_DEFAULT_REG_L) SetDefaultLChoice(REGULATORY);
         if (Control == LAND_MENU_DEFAULT_AP42_L) SetDefaultLChoice(AP42);

         CheckLandwinLMenu(hMenu, Control);
         return TRUE;

      case LAND_MENU_CODIS:
         SetLandfillType(CODISPOSAL);
         ChoicesToMenuSelections(hWnd);
         return TRUE;

      case LAND_MENU_NOCO:
         SetLandfillType(NOCODISPOSAL);
         ChoicesToMenuSelections(hWnd);
         return TRUE;

      case LAND_MENU_AIRTOXICS:
         GetATInfo(hWnd, hInst);
         return TRUE;

      case LAND_MENU_CLOSURE:
         GetClosureYear(hWnd, hInst);
         return TRUE;

      case LAND_MENU_GRAPHICS:
         if (ValidateData())
            {
            if ((ret = GetSubstanceSelection(hWnd, hInst)) > 0)
               CreateGraphic(CalculationModule(ret - 1));
            }
         return TRUE;

      case LAND_MENU_TEXT:
         if (ValidateData())
            {
```



```
            if ((ret = GetSubstanceSelection(hWnd, hInst)) > 0)
               CreateReport(CalculationModule(ret - 1));
            }
         return TRUE;

      case LAND_MENU_OPTIONS:
         GetOptions(hWnd, hInst);
         return TRUE;

      case LAND_MENU_ESTIMATE:
         RefuseEstimator(hWnd, hInst);
         return TRUE;

      case LAND_MENU_CONVERT:
         if (WinExec("UNITFW", SW_SHOWNORMAL) < 32)
            MessageBox(NULL, "Error Loading Program", "Exec Error",
                       MB_ICONSTOP | MB_OK);

         return TRUE;

      case LAND_MENU_TILE_VERTICAL:
         SendMessage(RetrieveClientWindow(), WM_MDITILE, MDITILE_VERTICAL, 0);
         return TRUE;

      case LAND_MENU_TILE_HORIZONTAL:
         SendMessage(RetrieveClientWindow(), WM_MDITILE, MDITILE_HORIZONTAL, 0);
         return TRUE;

      case LAND_MENU_CASCADE:
         SendMessage(RetrieveClientWindow(), WM_MDICASCADE, MDITILE_SKIPDISABLED
,
                     0);
         return TRUE;

      case LAND_MENU_ICONARRANGE:
         SendMessage(RetrieveClientWindow(), WM_MDIICONARRANGE, 0, 0);
         return TRUE;

      case LAND_MENU_HELP_INDEX:
         WinHelp(hWnd, "LANDWIN.HLP", HELP_CONTENTS, 0L);
         return TRUE;

      case LAND_MENU_ABOUT:
         DisplayAbout(hWnd, hInst);
         return TRUE;

      default:
         MessageBox(hWnd, "The selection you requested has not yet been" \
                          " implemented",
                          "Selection Not Available", MB_OK | MB_ICONSTOP);

         return TRUE;
      }
]


/* ---------------------------------------------------------------------- */
/*

   Procedure:  CheckLandwinDefaultMenu

   Purpose:    Handles the selection (and the check marks) for the
               defaults selection mode on the menu.

*/
```

```
/* ---------------------------------------------------------------- */


void CheckLandwinDefaultMenu(HMENU hMenu, WORD id)
{
   CheckMenuItem(hMenu, LAND_MENU_REGULATORY,      MF_UNCHECKED);
   CheckMenuItem(hMenu, LAND_MENU_AP42,            MF_UNCHECKED);
   CheckMenuItem(hMenu, LAND_MENU_SPECIFIC,        MF_UNCHECKED);

   if ((int) id != -1) CheckMenuItem(hMenu, id, MF_CHECKED);
}

/* ---------------------------------------------------------------- */
/*

   Procedure:  CheckLandwinKMenu

   Purpose:    Handles the selection (the check mark) for the K
               choices on the application menu.

*/
/* ---------------------------------------------------------------- */

void CheckLandwinKMenu(HMENU hMenu, WORD id)
{
   CheckMenuItem(hMenu, LAND_MENU_DEFAULT_REG_K,   MF_UNCHECKED);
   CheckMenuItem(hMenu, LAND_MENU_DEFAULT_AP42_K,  MF_UNCHECKED);
   CheckMenuItem(hMenu, LAND_MENU_USER_K,          MF_UNCHECKED);
   CheckMenuItem(hMenu, LAND_MENU_CALC_K,          MF_UNCHECKED);

   if ((int) id != -1) CheckMenuItem(hMenu, id, MF_CHECKED);
}

/* ---------------------------------------------------------------- */
/*

   Procedure:  CheckLandwinLMenu

   Purpose:    Handles the selection (the check mark) for the L
               choices on the application menu.

*/
/* ---------------------------------------------------------------- */



void CheckLandwinLMenu(HMENU hMenu, WORD id)
{
   CheckMenuItem(hMenu, LAND_MENU_DEFAULT_REG_L,   MF_UNCHECKED);
   CheckMenuItem(hMenu, LAND_MENU_DEFAULT_AP42_L,  MF_UNCHECKED);
   CheckMenuItem(hMenu, LAND_MENU_USER_L,          MF_UNCHECKED);

   if ((int) id != -1) CheckMenuItem(hMenu, id, MF_CHECKED);
}

/* ---------------------------------------------------------------- */
/*

   Procedure:  EnableLandwinKMenu

   Purpose:    Handles enabling and disabling the applicable
               K choices in response to changing the defaults
               mode.

*/
```



```
/* ---------------------------------------------------------------- */


void EnableLandwinKMenu(HMENU hMenu, WORD id)
{
   EnableMenuItem(hMenu, LAND_MENU_DEFAULT_REG_K,   MF_GRAYED);
   EnableMenuItem(hMenu, LAND_MENU_DEFAULT_AP42_K,  MF_GRAYED);
   EnableMenuItem(hMenu, LAND_MENU_USER_K,          MF_GRAYED);
   EnableMenuItem(hMenu, LAND_MENU_CALC_K,          MF_GRAYED);

   if (id == 0)
      {
      EnableMenuItem(hMenu, LAND_MENU_DEFAULT_REG_K,   MF_ENABLED);
      EnableMenuItem(hMenu, LAND_MENU_DEFAULT_AP42_K,  MF_ENABLED);
      EnableMenuItem(hMenu, LAND_MENU_USER_K,          MF_ENABLED);
      EnableMenuItem(hMenu, LAND_MENU_CALC_K,          MF_ENABLED);
      }
   else if ((int) id != -1) EnableMenuItem(hMenu, id, MF_ENABLED);
}


/* ---------------------------------------------------------------- */
/*

   Procedure:  EnableLandwinLMenu

   Purpose:    Handles enabling and disabling the applicable
               L choices in response to changing the defaults
               mode.


*/
/* ---------------------------------------------------------------- */


void EnableLandwinLMenu(HMENU hMenu, WORD id)
{
   EnableMenuItem(hMenu, LAND_MENU_DEFAULT_REG_L,   MF_GRAYED);
   EnableMenuItem(hMenu, LAND_MENU_DEFAULT_AP42_L,  MF_GRAYED);
   EnableMenuItem(hMenu, LAND_MENU_USER_L,          MF_GRAYED);

   if (id == 0)
      {
      EnableMenuItem(hMenu, LAND_MENU_DEFAULT_REG_L,   MF_ENABLED);
      EnableMenuItem(hMenu, LAND_MENU_DEFAULT_AP42_L,  MF_ENABLED);
      EnableMenuItem(hMenu, LAND_MENU_USER_L,          MF_ENABLED);
      }
   else if ((int) id != -1) EnableMenuItem(hMenu, id, MF_ENABLED);
}


/* ---------------------------------------------------------------- */
/*

   Procedure:  SetDefaultMode

   Purpose:    Sets the default selection choice in the memory
               data structure in response to user selection.

*/
/* ---------------------------------------------------------------- */



void SetDefaultMode(int Choice)
{
```

```
HWND ActiveWindow;
HGLOBAL hOperData;
OPERDATASTRUCT *OperData = NULL;

   if (Choice == AP42) DisplayAP42Warn();

   ActiveWindow = MDIGETACTIVE();
   if (ActiveWindow != NULL)
      {
      hOperData = MDIGETOPERDATA(ActiveWindow);
      if (hOperData != NULL)
         OperData = (OPERDATASTRUCT *) GlobalLock(hOperData);
      }
   else return;

   if (OperData != NULL) OperData->Choices.DefaultsMode = Choice;

   GlobalUnlock(hOperData);
   return;
}

/* ------------------------------------------------------------------ */
/*

   Procedure:  SetDefaultKChoice

   Purpose:    Sets the appropriate K selection choice in the
               memory data structure in response to the user
               selection.  Note also that the system displays a
               warning screen in response to the AP42 default
               mode.

*/
/* ------------------------------------------------------------------ */


void SetDefaultKChoice(int Choice)
{
HWND ActiveWindow;
HGLOBAL hOperData;
OPERDATASTRUCT *OperData = NULL;

   if (Choice == AP42) DisplayAP42Warn();

   ActiveWindow = MDIGETACTIVE();
   if (ActiveWindow != NULL)
      {
      hOperData = MDIGETOPERDATA(ActiveWindow);
      if (hOperData != NULL)
         OperData = (OPERDATASTRUCT *) GlobalLock(hOperData);
      }
   else return;

   if (OperData != NULL) OperData->Choices.kChoice = Choice;

   GlobalUnlock(hOperData);
   return;
}

/* ------------------------------------------------------------------ */
/*

   Procedure:  SetDefaultLChoice

   Purpose:    Sets the appropriate L selection choice in the
               memory data structure in response to the user
```



```
               selection.  Note also that the system displays a
               warning screen in response to the AP42 default
               mode.

*/
/* ------------------------------------------------------------------ */

void SetDefaultLChoice(int Choice)
{
HWND ActiveWindow;
HGLOBAL hOperData;
OPERDATASTRUCT *OperData;

   if (Choice == AP42) DisplayAP42Warn();

   ActiveWindow = MDIGETACTIVE();
   if (ActiveWindow != NULL)
      {
      hOperData = MDIGETOPERDATA(ActiveWindow);
      if (hOperData != NULL)
         OperData = (OPERDATASTRUCT *) GlobalLock(hOperData);
      }
   else return;

   if (OperData != NULL) OperData->Choices.LOChoice = Choice;

   GlobalUnlock(hOperData);
   return;
}


/* ------------------------------------------------------------------ */
/*

   Procedure:  SetDefaultNMOCChoice

   Purpose:    Sets the default NMOC selection in response to the
               user's choice on the compound selection dialog.

*/
/* ------------------------------------------------------------------ */



void SetDefaultNMOCChoice(int Choice)
{
HWND ActiveWindow;
HGLOBAL hOperData;
OPERDATASTRUCT *OperData;

   if (Choice == AP42) DisplayAP42Warn();

   ActiveWindow = MDIGETACTIVE();
   if (ActiveWindow != NULL)
      {
      hOperData = MDIGETOPERDATA(ActiveWindow);
      if (hOperData != NULL)
         OperData = (OPERDATASTRUCT *) GlobalLock(hOperData);
      }
   else return;

   if (OperData != NULL) OperData->Choices.NMOCChoice = Choice;

   GlobalUnlock(hOperData);
   return;
}
```

```
/* ------------------------------------------------------------------ */
/*

   Procedure:  SetLandfillType

   Purpose:    Tags the memory data structure with the appropriate
               landfill type selection in response to the user.

*/
/* ------------------------------------------------------------------ */


void SetLandfillType(int Choice)
{
HWND ActiveWindow;
HGLOBAL hOperData;
OPERDATASTRUCT *OperData;

   ActiveWindow = MDIGETACTIVE();
   if (ActiveWindow != NULL)
      {
      hOperData = MDIGETOPERDATA(ActiveWindow);
      if (hOperData != NULL)
         OperData = (OPERDATASTRUCT *) GlobalLock(hOperData);
      }
   else return;

   if (OperData != NULL) OperData->Choices.LandfillType = Choice;

   GlobalUnlock(hOperData);
   return;
}


/* ------------------------------------------------------------------ */
/*

   Procedure:  DefaultsToMenuSelection

   Purpose:    Handles the selection of the appropriate menu items
               in response to the user changing the default mode.

*/
/* ------------------------------------------------------------------ */


void DefaultsToMenuSelections(HWND hWnd)
{
HMENU hMenu;
WORD ControlL0, ControlK;
HWND ActiveWindow;
HGLOBAL hOperData;
OPERDATASTRUCT *OperData;

   ActiveWindow = MDIGETACTIVE();
   if (ActiveWindow != NULL)
      {
      hOperData = MDIGETOPERDATA(ActiveWindow);
      if (hOperData != NULL)
         OperData = (OPERDATASTRUCT *) GlobalLock(hOperData);
      }
   else return;
```



```
   if (OperData != NULL)
      {
      hMenu = GetMenu(hWnd);

/* Check and enable the appropriate entry on the Parameters Menu */

      if (OperData->Choices.DefaultsMode == REGULATORY)
         {
         ControlK  = LAND_MENU_DEFAULT_REG_K;
         ControlL0 = LAND_MENU_DEFAULT_REG_L;
         }
      else if (OperData->Choices.DefaultsMode == AP42)
         {
         ControlK  = LAND_MENU_DEFAULT_AP42_K;
         ControlL0 = LAND_MENU_DEFAULT_AP42_L;
         }
      else if (OperData->Choices.DefaultsMode == USER)
         {
         ControlK  = 0;
         ControlL0 = 0;
         }

      EnableLandwinKMenu(hMenu, ControlK);
      EnableLandwinLMenu(hMenu, ControlL0);

      if (ControlK == 0)
         {
         ControlK  = LAND_MENU_USER_K;
         ControlL0 = LAND_MENU_USER_L;
         }

      CheckLandwinKMenu(hMenu,  ControlK);
      CheckLandwinLMenu(hMenu,  ControlL0);

      SetDefaultKChoice(OperData->Choices.DefaultsMode);
      SetDefaultLChoice(OperData->Choices.DefaultsMode);
      SetDefaultNMOCChoice(OperData->Choices.DefaultsMode);

      GlobalUnlock(hOperData);
      }
}


/* ------------------------------------------------------------------ */
/*

   Procedure:  ChoicesToMenuSelections

   Purpose:    Handles syncronizing the menu items with the data
               in the memory structure.  This is useful when loading
               as study from disk or when switching MDI windows.

*/
/* ------------------------------------------------------------------ */


void ChoicesToMenuSelections(HWND hWnd)
{
HMENU hMenu;
WORD ControlL0, ControlK;
HWND ActiveWindow;
HGLOBAL hOperData;
OPERDATASTRUCT *OperData;

   ActiveWindow = MDIGETACTIVE();
```

```
   if (ActiveWindow != NULL)
      {
      hOperData = !!GETOPERDATA(ActiveWindow);
      if (hOperData != NULL)
         OperData = (OPERDATASTRUCT *) GlobalLock(hOperData);
      else return;
      }
   else return;

   if (OperData != NULL)
      {
      hMenu = GetMenu(hWnd);

/* Check and enable the appropriate entry on the Parameters Menu */

      if (OperData->Choices.DefaultsMode == REGULATORY)
         {
         CheckLandwinDefaultMenu(hMenu, LAND_MENU_REGULATORY);
         ControlK  = LAND_MENU_DEFAULT_REG_K;
         ControlL0 = LAND_MENU_DEFAULT_REG_L;
         }
      else if (OperData->Choices.DefaultsMode == AP42)
         {
         CheckLandwinDefaultMenu(hMenu, LAND_MENU_AP42);
         ControlK  = LAND_MENU_DEFAULT_AP42_K;
         ControlL0 = LAND_MENU_DEFAULT_AP42_L;
         }
      else if (OperData->Choices.DefaultsMode == USER)
         {
         CheckLandwinDefaultMenu(hMenu, LAND_MENU_SPECIFIC);
         ControlK  = 0;
         ControlL0 = 0;
         }

      EnableLandwinKMenu(hMenu, ControlK);
      EnableLandwinLMenu(hMenu, ControlL0);


      if (OperData->Choices.kChoice == REGULATORY)
         ControlK = LAND_MENU_DEFAULT_REG_K;
      else if (OperData->Choices.kChoice == AP42)
         ControlK = LAND_MENU_DEFAULT_AP42_K;
      else if (OperData->Choices.kChoice == USER)
         ControlK = LAND_MENU_USER_K;
      else ControlK = LAND_MENU_CALC_K;

      CheckLandwinKMenu(hMenu,  ControlK);

      if (OperData->Choices.L0Choice == REGULATORY)
         ControlL0 = LAND_MENU_DEFAULT_REG_L;
      else if (OperData->Choices.L0Choice == AP42)
         ControlL0 = LAND_MENU_DEFAULT_AP42_L;
      else ControlL0 = LAND_MENU_USER_L;

      CheckLandwinLMenu(hMenu,  ControlL0);

      if (OperData->Choices.LandfillType == NOCODISPOSAL)
         {
         CheckMenuItem(hMenu, LAND_MENU_NOCO,   MF_CHECKED);
         CheckMenuItem(hMenu, LAND_MENU_CODIS,  MF_UNCHECKED);
         }
      else
         {
         CheckMenuItem(hMenu, LAND_MENU_NOCO,   MF_UNCHECKED);
         CheckMenuItem(hMenu, LAND_MENU_CODIS,  MF_CHECKED);
         }
```



```
      GlobalUnlock(hOperData);
      }
}
```

```
// ============================================================================
//                                  LEDIT.C
// ============================================================================
// ============================================================================
//                         Allows editing of the L value.
// ============================================================================
// ============================================================================
//                             Revision History
// ============================================================================
//  09/30/94     Radian(RLB)              Initial Revision
// ============================================================================
#include <windows.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>

#include "landwin.h"
#include "str_ops.h"
#include "helpdef.h"

BOOL CALLBACK DialogProc(HWND hWndDlg, unsigned uiMsg, WORD wParam,
                         LONG lParam);


int GetLValue(HWND hWnd, HINSTANCE hInst)
{
FARPROC DlgProcInstance;
int ret_value;

   DlgProcInstance = MakeProcInstance(DialogProc, hInst);
   ret_value = DialogBox(hInst, "L_DIALOG", hWnd, DlgProcInstance);
   FreeProcInstance(DlgProcInstance);

   return ret_value;
}


#pragma argsused

static BOOL CALLBACK DialogProc(HWND hWndDlg, unsigned uiMsg, WORD wParam,
                                LONG lParam)
{
char string[30];
HWND ActiveWindow;
HGLOBAL hOperData = NULL;
OPERDATASTRUCT *OperData;

   switch (uiMsg)
      {
      case WM_INITDIALOG:
         CenterWindow(hWndDlg);

         ActiveWindow = SendMessage(RetrieveClientWindow(), WM_MDIGETACTIVE,
                                    0, 0);
         if (ActiveWindow != NULL)
            hOperData = SendMessage(ActiveWindow, AM_GETOPERHANDLE, 0, 0);

         if (hOperData != NULL)
            OperData = (OPERDATASTRUCT *) GlobalLock(hOperData);


         sprintf(string, "%.2f", OperData->Model.10);
         SetDlgItemText(hWndDlg, IDC_VALUEEDIT, (LPCSTR) string);
         CustomizeControl(hWndDlg, IDC_VALUEEDIT, (FARPROC) FPNumberEditProc);

         if (hOperData != NULL) GlobalUnlock(hOperData);
```



```
         return TRUE;

      case WM_COMMAND:
         switch (LOWORD(wParam))
            {
            case IDC_OK:
               GetDlgItemText(hWndDlg, IDC_VALUEEDIT, (LPSTR) string,
                              sizeof(string));

               ActiveWindow = SendMessage(RetrieveClientWindow(),
                                          WM_MDIGETACTIVE, 0, 0);
               if (ActiveWindow != NULL)
                  hOperData = SendMessage(ActiveWindow, AM_GETOPERHANDLE, 0, 0);

               if (hOperData != NULL)
                  OperData = (OPERDATASTRUCT *) GlobalLock(hOperData);

               OperData->Model.10 = strtod(string, NULL);

               if (hOperData != NULL) GlobalUnlock(hOperData);

               EndDialog(hWndDlg, TRUE);
               return TRUE;

            case IDC_CANCEL:
               EndDialog(hWndDlg, FALSE);
               return TRUE;

            case IDC_HELP:
               WinHelp(hWndDlg, "LANDWIN.HLP", HELP_CONTEXT, ParametersUser);
               return TRUE;

            case IDC_VALUEEDIT:
               return FALSE;

            default:
               break;
            }
      }

   return FALSE;
}
```

```
// ==========================================================================
//                                LOADDEF.C
// ==========================================================================
// ==========================================================================
//  Handles the loading of resource/ini information and saving settings.
// ==========================================================================
// ==========================================================================
//                            Revision History
// ==========================================================================
//  09/30/94    Radian(RLB)             Initial Revision
// ==========================================================================
#include <windows.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dos.h>

#include "landwin.h"
#include "landoper.h"
#include "datadef.h"
#include "str_ops.h"

double ProfileDouble(char *ProfileString, int ResourceIdDefault);

static char *ProfileSection = "LAEEM";

void LoadModelDefaults(HWND hWnd)
{
int ATCount;
struct date date;
HGLOBAL hOperData;
OPERDATASTRUCT *OperData;
AIRTOXIC AT, *ATPtr;
double ResourceDouble(int ResourceID);
double ProfileDouble(char *ProfileString, int ResourceId);
void LoadModelLimits(void);
void ResourceAT(int ResourceInt, AIRTOXIC *AT);

   hOperData = SendMessage(hWnd, AM_GETOPERHANDLE, 0, 0);
   if (hOperData == NULL) return;

   OperData = (OPERDATASTRUCT *) GlobalLock(hOperData);

/* Model Parameters (These do not depend on defaults mode) */

   OperData->Model.PercentCH4 = ResourceDouble(DEFAULT_CH4);
   OperData->Model.PercentCO2 = ResourceDouble(DEFAULT_CO2);
   OperData->Model.Temp       = ResourceDouble(DEFAULT_TEMP);
   OperData->Model.Pressure   = ResourceDouble(DEFAULT_PRESSURE);
   OperData->Model.MoleWt     = ResourceDouble(DEFAULT_MOLE_WT);

/* Method 2E Parameters */

   OperData->Method2E.WellDepth   = ResourceDouble(DEFAULT_WELL_DEPTH);
   OperData->Method2E.ROI         = ResourceDouble(DEFAULT_INFLUENCE_RADIUS);
   OperData->Method2E.Density     = ResourceDouble(DEFAULT_REFUSE_DENSITY);
   OperData->Method2E.Fraction    = ResourceDouble(DEFAULT_DECOMP_FRACTION);
   OperData->Method2E.FlowRate    = ResourceDouble(DEFAULT_FLOW_RATE);
   OperData->Method2E.RefuseAge   = ResourceDouble(DEFAULT_REFUSE_AGE);

/* Initialize the Operational Parameters */

/* getdate is not valid for Win32 */

   getdate(&date);
```



```
   OperData->Parameters.YearOpened = date.da_year - 10;
   OperData->Parameters.YearCurrent = date.da_year;

   if (OperData->Periods.hPeriod != NULL)
      GlobalFree(OperData->Periods.hPeriod);

   OperData->Periods.PeriodCount = OperData->Parameters.YearCurrent
                                 - OperData->Parameters.YearOpened + 1;
   OperData->Periods.hPeriod = (HGLOBAL) GlobalAlloc(GHND,
                               sizeof(double) * OperData->Periods.PeriodCount);

/* Air Toxic Stubs for later */

   if (OperData->AirToxics.hAirToxic != NULL)
      GlobalFree(OperData->AirToxics.hAirToxic);

   for (ATCount = 0 ; ; ATCount++)
      {
      ResourceAT(AIRTOXICSRC + ATCount, &AT);

      if (! strcmp(AT.Name, "END")) break;
      }

   OperData->AirToxics.hAirToxic = (HGLOBAL) GlobalAlloc(GHND,
                                   sizeof(AIRTOXIC) * ATCount);
   OperData->AirToxics.AirToxicCount = ATCount;

   if (OperData->AirToxics.hAirToxic != NULL)
      {
      ATPtr = (AIRTOXIC *) GlobalLock(OperData->AirToxics.hAirToxic);

      for (ATCount = 0 ; ATCount < OperData->AirToxics.AirToxicCount ; ATCount++
)
         ResourceAT(AIRTOXICSRC + ATCount, ATPtr + ATCount);

      GlobalUnlock(OperData->AirToxics.hAirToxic);
      }

   GlobalUnlock(hOperData);
}



void SetModelDefaults(HWND hWnd)
{
HGLOBAL hOperData;
OPERDATASTRUCT *OperData;
double ResourceDouble(int ResourceID);
double ProfileDouble(char *ProfileString, int ResourceId);

   hOperData = SendMessage(hWnd, AM_GETOPERHANDLE, 0, 0);
   if (OperData == NULL) return;

   OperData = (OPERDATASTRUCT *) GlobalLock(hOperData);

   if (OperData->Choices.L0Choice == REGULATORY)
      OperData->Model.l0    = ResourceDouble(DEFAULT_REGULATORY_L0);
   else if (OperData->Choices.L0Choice == AP42)
      OperData->Model.l0    = ResourceDouble(DEFAULT_AP42_L0);
   else
      OperData->Model.l0    = Limits.UserL0;

   if (OperData->Choices.kChoice == REGULATORY)
      OperData->Model.k     = ResourceDouble(DEFAULT_REGULATORY_K);
   else if (OperData->Choices.kChoice == AP42)
```

```
      OperData->Model.k      = ResourceDouble(DEFAULT_AP42_K);
   else
      OperData->Model.k      = Limits.Userk;

   if (OperData->Choices.NMOCChoice == REGULATORY)
      OperData->Model.NMOC   = ResourceDouble(DEFAULT_REGULATORY_NMOC);
   else if (OperData->Choices.NMOCChoice == AP42)
      {
      if (OperData->Choices.LandfillType == CODISPOSAL)
         OperData->Model.NMOC   = ResourceDouble(DEFAULT_AP42_NMOC_CODIS);
      else
         OperData->Model.NMOC   = ResourceDouble(DEFAULT_AP42_NMOC_NOCO);
      }
   else
      OperData->Model.NMOC      = Limits.UserNMOC;

   GlobalUnlock(hOperData);

}



void LoadModelLimits(void)
{
char string[50], Resource[50];

/* Read the mode defaults section */

   GetProfileString(ProfileSection, "DefaultsMode", "Regulatory",
                    string, 49);

   strupper(string, string);
   if (! strcmp("REGULATORY", string))
      Limits.DefaultsMode = REGULATORY;
   else if (! strcmp("AP-42", string))
      Limits.DefaultsMode = AP42;
   else if (! strcmp("USER", string))
      Limits.DefaultsMode = USER;

   GetProfileString(ProfileSection, "LandfillType", "NOCODISPOSAL",
                    string, 49);

   strupper(string, string);
   if (! strcmp("CODISPOSAL", string))
      Limits.LandfillType = CODISPOSAL;
   else if (! strcmp("NOCODISPOSAL", string))
      Limits.LandfillType = NOCODISPOSAL;
   else if (! strcmp("USER", string))
      Limits.LandfillType = UNKNOWN;


   Limits.Userk    = ProfileDouble("USER_K",    DEFAULT_REGULATORY_K);
   Limits.UserL0   = ProfileDouble("USER_L0",   DEFAULT_REGULATORY_L0);
   Limits.UserNMOC = ProfileDouble("USER_NMOC", DEFAULT_REGULATORY_NMOC);

   /* Read the max years section */

   Limits.MaxYears = GetProfileInt(ProfileSection, "MaxYears", 200);
}



void SaveModelLimits(void)
{
char string[30];
```



```
   switch (Limits.DefaultsMode)
      {
      case REGULATORY:
         strcpy(string, "Regulatory");
         break;

      case AP42:
         strcpy(string, "AP-42");
         break;

      case USER:
         strcpy(string, "User");
         break;
      }

   WriteProfileString(ProfileSection, "DefaultsMode", string);

   switch (Limits.LandfillType)
      {
      case NOCODISPOSAL:
         strcpy(string, "NoCodisposal");
         break;

      case CODISPOSAL:
         strcpy(string, "Codisposal");
         break;

      case UNKNOWN:
         strcpy(string, "Unknown");
         break;
      }

   WriteProfileString(ProfileSection, "LandfillType", string);

   sprintf(string, "%d", Limits.MaxYears);
   WriteProfileString(ProfileSection, "MaxYears", string);

   sprintf(string, "%0.6g", Limits.UserL0);
   WriteProfileString(ProfileSection, "User_l0", string);

   sprintf(string, "%0.6g", Limits.Userk);
   WriteProfileString(ProfileSection, "User_k", string);

   sprintf(string, "%0.6g", Limits.UserNMOC);
   WriteProfileString(ProfileSection, "User_NMOC", string);

/*  Flush the profile strings */

   WriteProfileString(NULL, NULL, NULL);
}


void ResourceAT(int ResourceInt, AIRTOXIC *AT)
{
char string[80];

   LoadString(RetrieveAppInstance(), ResourceInt, (LPSTR) string,
              sizeof(string) - 1);

   strcpy(AT->Name, strtok(string, ";"));

   AT->MoleWt    = strtod(strtok(NULL, ";"), NULL);
   AT->EmFact[0] = strtod(strtok(NULL, ";"), NULL);
   AT->EmFact[1] = strtod(strtok(NULL, ";"), NULL);
}
```

```
double ResourceDouble(int ResourceInt)
{
char string[50];
double value;

   LoadString(RetrieveAppInstance(), ResourceInt, (LPSTR) string,
              sizeof(string) - 1);
   return strtod(string, NULL);
}


double ProfileDouble(char *ProfileString, int ResourceIdDefault)
{
char string[50], resource[50];

   LoadString(RetrieveAppInstance(), ResourceIdDefault, (LPSTR) resource,
              sizeof(resource));
   GetProfileString(ProfileSection, ProfileString, resource,
                    string, sizeof(string));

   return strtod(string, NULL);
}
```

```
// ========================================================================
//                              MAKEGPH.C
// ========================================================================
// ========================================================================
//                    Creates a graphical output window.
// ========================================================================
// ========================================================================
//                            Revision History
// ========================================================================
//  09/30/94    Radian(RLB)             Initial Revision
// ========================================================================
#include <stdio.h>
#include <string.h>
#include <windows.h>

#include "landwin.h"


extern char StudyName[MAXPATH];

void CreateGraphic(HGLOBAL hResults)
{
HWND ActiveWindow, hWnd;
HGLOBAL hOperData;
OPERDATASTRUCT *OperData;

   ActiveWindow = MDIGETACTIVE();
   if (IsWindow(ActiveWindow)) hOperData = MDIGETOPERDATA(ActiveWindow);

   if (hOperData != NULL)
      OperData = (OPERDATASTRUCT *) GlobalLock(hOperData);

   if (OperData != NULL)
      {
      sprintf(StudyName, "Graphics: %s", OperData->Filename);

      hWnd = CreateGraphClientWindow(StudyName,
                           CW_USEDEFAULT, CW_USEDEFAULT,
                           CW_USEDEFAULT, CW_USEDEFAULT,
                           WS_OVERLAPPEDWINDOW | WS_VISIBLE);

      SetWindowLong(hWnd, 0, (long) hOperData);
      SetWindowLong(hWnd, 8, (long) hResults);

      if (hWnd != NULL)
         {
         ShowWindow(hWnd, SW_SHOWNORMAL);
         SendMessage(RetrieveClientWindow(), WM_MDIACTIVATE, hWnd, 0);
         }

      GlobalUnlock(hOperData);
      }
}
```

```
// ========================================================================
//                                MAKERPT.C
// ========================================================================
// ========================================================================
//                        Makes a report output window
// ========================================================================
// ========================================================================
//                            Revision History
// ========================================================================
//  09/30/94    Radian(RLB)             Initial Revision
// ========================================================================

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

#include "landwin.h"


extern char StudyName[MAXPATH];

void CreateReport(HGLOBAL hResults)
{
HWND ActiveWindow, hWnd;
HGLOBAL hOperData;
OPERDATASTRUCT *OperData;

   ActiveWindow = MDIGETACTIVE();
   if (IsWindow(ActiveWindow)) hOperData = MDIGETOPERDATA(ActiveWindow);

   if (hOperData != NULL)
      OperData = (OPERDATASTRUCT *) GlobalLock(hOperData);

   if (OperData != NULL)
      {
      sprintf(StudyName, "Text Report: %s", OperData->Filename);

      hWnd = CreateTextClientWindow(StudyName,
                              CW_USEDEFAULT, CW_USEDEFAULT,
                              CW_USEDEFAULT, CW_USEDEFAULT,
                              WS_OVERLAPPEDWINDOW | WS_VISIBLE
                              | WS_VSCROLL | WS_HSCROLL);

      SetWindowLong(hWnd, 0, (long) hOperData);
      SetWindowLong(hWnd, 8, (long) hResults);

      if (hWnd != NULL)
         {
         ShowWindow(hWnd, SW_SHOWNORMAL);
         SendMessage(RetrieveClientWindow(), WM_MDIACTIVATE, hWnd, 0);

         SendMessage(hWnd, AM_VALUESCHANGED, 0, 0);
         }

      GlobalUnlock(hOperData);
      }
}
```

```
// ========================================================================
//                                  OPER.C
// ========================================================================
// ========================================================================
//      Handles the creation and display of the operational paramaters
//      MDI client.
// ========================================================================
// ========================================================================
//                             Revision History
// ========================================================================
//  09/30/94    Radian(RLB)              Initial Revision
// ========================================================================

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

#include "landwin.h"
#include "landoper.h"

#include "str_ops.h"


/* Local Function Declarations */

void FreeOperMemory(HWND hWnd);

void HighlightCell(HWND hWnd, HDC hDC, int CellCurrent);
void UnHighlightCell(HWND hWnd, HDC hDC, int CellCurrent);
void UpdateEntry(HWND hWnd, HDC hDC, int Cell);
void MarkSelection(HWND hWnd, HDC hDC, int Mark);

BOOL AllocOperMemory(HWND hWnd);

HWND CreateOperClientWindow(char *title, int x, int y, int width, int height,
                            DWORD style)
{
MDICREATESTRUCT MDIcs;
HWND hWnd;

/* Set up the MDI Structure to create the client window.  lParam contains */
/* an id that is stored in the window that indicates the type of MDI */
/* window this is and which WndProc needs to be invoked. */

   MDIcs.szClass = "OPERWINDOW";
   MDIcs.szTitle = title;
   MDIcs.hOwner = RetrieveAppInstance();
   MDIcs.x = x;
   MDIcs.y = y;
   MDIcs.cx = width;
   MDIcs.cy = height;
   MDIcs.style = style;
   MDIcs.lParam = 0;

   hWnd = SendMessage(RetrieveClientWindow(), WM_MDICREATE, 0, (LPARAM) &MDIcs);

   return hWnd;
}



void RegisterOperClass(void)
{
```



```
WNDCLASS WndClass;
long FAR PASCAL OperClientWndProc(HWND hWnd, WORD iMessage, WPARAM wParam,
                                  LPARAM lParam);

   WndClass.cbClsExtra = 0;
   WndClass.cbWndExtra = sizeof(long);
   WndClass.hbrBackground = GetStockObject(WHITE_BRUSH);
   WndClass.hCursor = LoadCursor(NULL, IDC_ARROW);
   WndClass.hIcon = NULL;
   WndClass.hInstance = RetrieveAppInstance();
   WndClass.lpfnWndProc = (WNDPROC) OperClientWndProc;
   WndClass.lpszClassName = "LANDOPERCLNT";
   WndClass.lpszMenuName = NULL;
   WndClass.style = CS_HREDRAW | CS_VREDRAW;

   RegisterClass(&WndClass);
}



void UnregisterOperClass(void)
{
   UnregisterClass("LANDOPERCLNT", RetrieveAppInstance());
}



long FAR PASCAL OperWndProc(HWND hWnd, WORD iMessage, WPARAM wParam,
                            LPARAM lParam)
{
TEXTMETRIC tm;
HWND Titles, Tool;
HWND hWndSrc, hWndDest;
HDC hDC;
RECT r;
OPERDATASTRUCT *OperData;
double *Period;
int WindowWidth, WindowHeight;
void DestroyAllRelations(HWND hOperWind);
void InformAllRelations(HWND hOperWind, UINT Message, WPARAM wParam,
                        LPARAM lParam);

     switch (iMessage)
        {
        case WM_CREATE:
           hDC = GetDC(hWnd);
           GetTextMetrics(hDC, &tm);

           GetClientRect(hWnd, &r);
           CreateWindow("LANDOPERCLNT", "", WS_CHILD | WS_VISIBLE | WS_VSCROLL,
                        0, 2 * tm.tmHeight + 1,
                        r.right - r.left, r.bottom - r.top - 2 * tm.tmHeight - 1
,
                        hWnd, NULL, RetrieveAppInstance(), NULL);

           Titles = CreateWindow("LANDCNT", "",
                                 WS_CHILD | WS_VISIBLE | CNT_SIZEX,
                                 0, 0, 30, 2 * tm.tmHeight, hWnd, NULL,
                                 RetrieveAppInstance(), NULL);
           CreateWindow("BorStatic", "Year", WS_CHILD | WS_VISIBLE,
                        tm.tmMaxCharWidth, tm.tmHeight / 2,
                        5 * tm.tmMaxCharWidth, tm.tmHeight, Titles, NULL,
                        RetrieveAppInstance(), NULL);
           CreateWindow("BorStatic", "Acceptance Rate (Mg)", WS_CHILD | WS_VISIB
LE,
                        tm.tmMaxCharWidth * 6, tm.tmHeight / 2,
```

```
                        20 * tm.tmMaxCharWidth, tm.tmHeight, Titles, NULL,
                        RetrieveAppInstance(), NULL);
            CreateWindow("BorStatic", "Refuse In Place (Mg)", WS_CHILD | WS_VISIB
LE,
                        tm.tmMaxCharWidth * 18, tm.tmHeight / 2,
                        18 * tm.tmMaxCharWidth, tm.tmHeight, Titles, NULL,
                        RetrieveAppInstance(), NULL);

            ReleaseDC(hWnd, hDC);
            return 0;

         case WM_DESTROY:
            DestroyAllRelations(hWnd);
            PostMessage(RetrieveFrameWindow(), AM_WINDOWACTIVE, 0, 0L);
            break;

         case WM_SIZE:
            hDC = GetDC(hWnd);
            GetTextMetrics(hDC, &tm);

            WindowHeight = HIWORD(lParam);
            WindowWidth = LOWORD(lParam);

            MoveWindow(GetWindow(hWnd, GW_CHILD), 0, 2 * tm.tmHeight + 1,
                       WindowWidth, WindowHeight - 2 * tm.tmHeight - 1,
                       TRUE);

            ReleaseDC(hWnd, hDC);
            break;

         case WM_MDIACTIVATE:
            SendMessage(RetrieveFrameWindow(), AM_WINDOWACTIVE, 0, 0);

            Tool = SendMessage(RetrieveFrameWindow(), AM_GETACTIVETOOL,
                               wParam, lParam);

            if (LOWORD(lParam) == hWnd)
               {
               SendMessage(GetWindow(hWnd, GW_CHILD), AM_MDIRESTORESTATE, 0, 0);

               if (Tool != NULL)
                  {
                  SendMessage(GetWindow(Tool, GW_CHILD), AM_FLUSHFIELDS, 0,
                                                                        HIWORD
(hWnd));
                  SendMessage(GetWindow(Tool, GW_CHILD), AM_UPDATEFIELDS, 0,
                              LOWORD(lParam));
                  }
               }
            else if (HIWORD(lParam) == hWnd)
               {
               SendMessage(GetWindow(hWnd, GW_CHILD), AM_MDISAVESTATE, 0, 0);
               if (Tool != NULL) SetFocus(Tool);
               }

            return FALSE;

         case AM_GETCURRENTENTRY:
            return SendMessage(GetWindow(hWnd, GW_CHILD), iMessage,
                               wParam, lParam);

                  case AM_PERIODCHANGED:
            return SendMessage(GetWindow(hWnd, GW_CHILD), iMessage,
                               wParam, lParam);

         case AM_CUT:
```



```
         case AM_PASTE:
            return SendMessage(GetWindow(hWnd, GW_CHILD), iMessage,
                               wParam, lParam);

         case AM_VALUESCHANGED:
            SendMessage(GetWindow(hWnd, GW_CHILD), AM_VALUESCHANGED, wParam, 0);
            InformAllRelations(hWnd, AM_VALUESCHANGED, 0, 0);
            return TRUE;

         case AM_GETOPERHANDLE:
            return (long) GetWindowLong(GetWindow(hWnd, GW_CHILD), 0);


                  case WM_KEYUP:
            return FALSE;

         case WM_KEYDOWN:
            switch (wParam)
               {
               case VK_UP:
               case VK_DOWN:
               case VK_PRIOR:
               case VK_NEXT:
                  return SendMessage(GetWindow(hWnd, GW_CHILD), iMessage,
                                     wParam, lParam);

               default:
                  Tool = SendMessage(RetrieveFrameWindow(), AM_GETACTIVETOOL,
                                     wParam, lParam);
                  if ((Tool != NULL) && (IsWindow(Tool)))
                                                  return SendMessage(Tool, iMess
age, wParam, lParam);
                  break;
               }

            break;
         }

      return(DefMDIChildProc(hWnd, iMessage, wParam, lParam));
}



// This window procedure handles the nuts and bolts of display of the
// operational parameters screen.  The control variables it tracks are
//
// CELLOFFSET = The position into the array at which the display begins
// CELLINDEX  = A pointer to the currently selected year
// CELLCOUNT  = The number of displayed cells in the window.
//
// So, the absolute position of the currently selected year is
// CELLOFFSET + CELLINDEX
//
// CELLOFFSET and CELLINDEX are zero based



long FAR PASCAL OperClientWndProc(HWND hWnd, WORD iMessage, WPARAM wParam,
                                  LPARAM lParam)
{
HWND Tool, hWndSrc, hWndDest;
HDC hDC;
PAINTSTRUCT lpps;
LOGFONT *plf;
HFONT hOldFnt, hFnt;
```

```
TEXTMETRIC tm;
OPERDATASTRUCT *OperData;
HANDLE hData;
HGLOBAL hCut;
char *Text;
char YearString[5], RefuseString[30];
int WindowWidth, WindowHeight;
int Width, Height;
int CellTempIndex, CellTempOffset;
LRESULT ret;
RECT r;
int i, j = 0;
int x, y;
double *Period;
void ScrollUp(HWND hWnd);
void ScrollDown(HWND hWnd);
void PageUp(HWND hWnd);
void PageDown(HWND hWnd);
HGLOBAL CutPeriods(HWND hWnd, int Mark);

     if ((iMessage != WM_DESTROY) && (iMessage != WM_CREATE))
         if (GetWindowLong(hWnd, 0) != NULL)
            OperData = (OPERDATASTRUCT *) GlobalLock(GetWindowLong(hWnd, 0));

     switch (iMessage)
        {
        case WM_DESTROY:
           FreeOperMemory(hWnd);
           OperData = NULL;
                        break;

                 case WM_CREATE:
           if (! AllocOperMemory(hWnd)) ret =  -1;
                        else ret = 0;
           break;

        case WM_KEYDOWN:
           if (OperData == NULL)
              {
              ret = TRUE;
                                 break;
              }

           switch (wParam)
              {
              case VK_UP:
                                       CellTempOffset = OperData->CellOffset;
                 CellTempIndex = OperData->CellIndex;

                 if (!(OperData->CellOffset + OperData->CellIndex))
                                              {

                    ret = TRUE;
                    break;
                    }

                 OperData->CellOffset--;
                 if (OperData->CellOffset < 0)
                                             {
                    OperData->CellOffset = 0;
                                             OperData->CellIndex--;

                    if (OperData->CellIndex < 0) OperData->CellIndex = 0;
                    }

                 if (OperData->CellIndex == CellTempIndex)
                    {
```



```
                    hDC = GetDC(hWnd);
                                                      UnHighlightCell(hWnd, hDC, Cel
lTempOffset);
                    HighlightCell(hWnd, hDC, OperData->CellOffset);
                    ReleaseDC(hWnd, hDC);
                    }
                 else InvalidateRect(hWnd, NULL, TRUE);

                 ret = TRUE;
                                       break;

              case VK_DOWN:
                 CellTempOffset = OperData->CellOffset;
                 CellTempIndex = OperData->CellIndex;

                                       if ((OperData->CellOffset + OperData->
CellIndex)
                       >= (OperData->Periods.PeriodCount - 1))
                    {
                    ret = TRUE;
                                                      break;
                    }

                                       OperData->CellOffset++;
                 if (OperData->CellOffset >= OperData->CellCount)
                    {
                    OperData->CellOffset--;
                                                      OperData->CellIndex++;
                    }

                 if (OperData->CellIndex == CellTempIndex)
                    {
                    hDC = GetDC(hWnd);
                                                      UnHighlightCell(hWnd, hDC, Cel
lTempOffset);
                    HighlightCell(hWnd, hDC, OperData->CellOffset);
                    ReleaseDC(hWnd, hDC);
                    }
                                          else InvalidateRect(hWnd, NULL, TRUE);

                 ret = TRUE;
                 break;

              case VK_PRIOR:
                 PageUp(hWnd);

                 ret = TRUE;
                 break;

              case VK_NEXT:
                 PageDown(hWnd);

                                          ret = TRUE;
                 break;

                                    default:
                 ret = FALSE;
                 break;
              }

        break;

                 case WM_VSCROLL:
        if (OperData == NULL)
           {
           ret = TRUE;
```

```
            break;
            }

         if (OperData->Mark) OperData->Mark = 0;

         switch (wParam)
                              {
            case SB_BOTTOM:
               OperData->CellIndex = OperData->Periods.PeriodCount - 1;
               OperData->CellOffset = 0;

               SetScrollPos(hWnd, SB_VERT, OperData->CellCount, TRUE);
               InvalidateRect(hWnd, NULL, TRUE);
                                    break;

            case SB_LINEDOWN:
                                    ScrollDown(hWnd);
               break;

                              case SB_LINEUP:
                                    ScrollUp(hWnd);
               break;

                              case SB_PAGEDOWN:
               PageDown(hWnd);
               break;

            case SB_PAGEUP:
               PageUp(hWnd);
               break;

            case SB_THUMBTRACK:
               OperData->CellIndex = LOWORD(lParam);
               if ((OperData->CellIndex + OperData->CellOffset)
                        > OperData->Periods.PeriodCount)
                  OperData->CellOffset = 0;

                                    if ((OperData->CellIndex + OperData->C
ellOffset) < 0)
                   OperData->CellOffset = 0;

                                    SetScrollPos(hWnd, SB_VERT, OperData->
CellIndex, FALSE);

               InvalidateRect(hWnd, NULL, TRUE);
               break;

            case SB_THUMBPOSITION:
               OperData->CellIndex = LOWORD(lParam);
                                    if ((OperData->CellIndex + OperData->C
ellOffset)
                           > OperData->Periods.PeriodCount)
                  OperData->CellOffset = 0;

               if ((OperData->CellIndex + OperData->CellOffset) < 0)
                  OperData->CellOffset = 0;

                                    SetScrollPos(hWnd, SB_VERT, OperData->
CellIndex, TRUE);

               InvalidateRect(hWnd, NULL, TRUE);

               break;
            }

         ret = TRUE;
```



```
         break;

               case AM_PERIODCHANGED:
         if (OperData == NULL)
            {
            ret = TRUE;
            break;
            }
// Mod 08/15/95 - Added support for dynamic periods - RLB

         if ((OperData->CellOffset + OperData->CellIndex)
               >= OperData->Periods.PeriodCount)
                                    {
             OperData->CellIndex = OperData->Periods.PeriodCount
                                    - OperData->CellOffset - 1;
             if (OperData->CellIndex < 0)
                {
                OperData->CellOffset += OperData->CellIndex;
                OperData->CellIndex = 0;
                                          if (OperData->CellOffset < 0) Op
erData->CellOffset = 0;
                }
             }

// End 08/15/95 Mod

                           if (OperData->Periods.PeriodCount <= OperData->CellCou
nt)
                                 SetScrollRange(hWnd, SB_VERT, 0, 0, TRUE);
         else
           {
                                 SetScrollRange(hWnd, SB_VERT, 0, OperData->Peri
ods.PeriodCount
                                       - OperData->CellCount, FALSE);
           SetScrollPos(hWnd, SB_VERT, OperData->CellIndex, TRUE);
           }

         InvalidateRect(hWnd, NULL, TRUE);
         ret = TRUE;
                           break;

      case AM_VALUESCHANGED:
         if (OperData == NULL)
            {
            ret = TRUE;
                              break;
                              }

         hDC = GetDC(hWnd);
                           if ((int) wParam == -1)
            {
            CellTempIndex = OperData->Periods.PeriodCount
                              - OperData->CellIndex;
//            for (i = OperData->CellOffset ;
                                    for (i = OperData->CellIndex ;
                                          (i <= OperData->CellCount) && (
i < CellTempIndex) ;
                                                i++) UpdateEntry(hWnd, h
DC, i);
            }
                           else UpdateEntry(hWnd, hDC, (int) wParam);
         ReleaseDC(hWnd, hDC);
         ret = TRUE;
         break;

               case AM_GETCURRENTENTRY:
```

```
            if (OperData == NULL)
               {
                              ret = -1;
               break;
               }

                           ret = OperData->CellIndex + OperData->CellOffset;
                           break;

                  case WM_SIZE:
                           if (OperData == NULL)
                                  {
                                  ret = FALSE;
                                  break;
                                  }

                           hDC = GetDC(hWnd);

                           GetTextMetrics(hDC, &tm);

                           WindowHeight = HIWORD(lParam);

// Mod 08/15/95 - Added -1 to calc of CellCount - RLB

                           OperData->CellCount = WindowHeight / (tm.tmHeight + 4)
 - 1;

// End Mod


                           if (OperData->Periods.PeriodCount <= OperData->CellCou
nt)
                                  {
                                  OperData->CellOffset += OperData->CellIndex;
                                  OperData->CellIndex = 0;
                                  SetScrollRange(hWnd, SB_VERT, 0, 0, TRUE);
                                  }
                           else
                                  {
                                  if (OperData->CellOffset > OperData->CellCount)
                                         {
                                         CellTempIndex = OperData->CellOffset -
OperData->CellCount + 1;
                                         OperData->CellIndex += CellTempIndex;
                                         OperData->CellOffset -= CellTempIndex;
                                         }


                                  SetScrollRange(hWnd, SB_VERT, 0, OperData->Peri
ods.PeriodCount
                                 - OperData->CellCount, FALSE);
                                  SetScrollPos(hWnd, SB_VERT, OperData->CellIndex
, TRUE);
                                  }

                           ReleaseDC(hWnd, hDC);

                           InvalidateRect(hWnd, NULL, TRUE);

                           Tool = SendMessage(RetrieveFrameWindow(), AM_GETACTIVE
TOOL,
wParam, lParam);


                           if (Tool != NULL)
                                  {
                                  SendMessage(GetWindow(Tool, GW_CHILD), AM_ENTR
```



```
YSELECTED,
                                                                  OperData->CellIn
dex + OperData->CellOffset, 0);

                                  ret = TRUE;
                                  }
                           else ret = FALSE;

                  break;

// Begins a drag select.  Capture all mouse input so we can be sure
// when the drag is over.

                  case WM_LBUTTONDOWN:
                           ret = TRUE;

                  if (OperData == NULL) break;
                           if ((hDC = GetDC(hWnd)) == NULL) break;
                  if (! GetTextMetrics(hDC, &tm)) break;

                  CellTempOffset = HIWORD(lParam) / (tm.tmHeight + 6);

// Changed to >= RLB 03/24/95

                  if ((OperData->CellIndex + CellTempOffset)
                           >= (OperData->Parameters.YearCurrent
                             - OperData->Parameters.YearOpened + 1))
                     {
                     OperData->Mark = -1;
                     ReleaseDC(hWnd, hDC);
                                  ret = TRUE;
                                  break;
                     }

                           OperData->Mark = CellTempOffset + OperData->CellIndex;
                  OperData->CellOffset = CellTempOffset;

                  SetCapture(hWnd);
                           if (GetCapture() != hWnd)
                     {
                     ReleaseDC(hWnd, hDC);
                     ret = FALSE;
                     break;
                     }

                  ReleaseDC(hWnd, hDC);

                           ret = TRUE;
                           break;

          case WM_MOUSEMOVE:
                           if (OperData == NULL)
                     {
                     ret = TRUE;
                     break;
                                  }

                  if (GetCapture() != hWnd)
                     {
                     ret = FALSE;
                     break;
                     }

                  if (wParam & MK_LBUTTON)
                                  {
                                  hDC = GetDC(hWnd);
```

```
            GetClientRect(hWnd, &r);

            if ((int) HIWORD(lParam) < 0)
               {
               CellTempOffset = OperData->CellOffset;
                                    CellTempIndex = OperData->CellIndex;

               if (OperData->CellOffset + OperData->CellIndex)
                  {
                  OperData->CellOffset--;
                  if (OperData->CellOffset < 0)
                     {
                     OperData->CellOffset = 0;
                     OperData->CellIndex--;

                                                         if (OperData->CellInde
x < 0) OperData->CellIndex = 0;
                     }
                  }

               if (OperData->CellIndex == CellTempIndex)
                  MarkSelection(hWnd, hDC, OperData->Mark);
               else InvalidateRect(hWnd, NULL, TRUE);
                                    }
            else if (HIWORD(lParam) > r.bottom)
               {
               CellTempOffset = OperData->CellOffset;
               CellTempIndex = OperData->CellIndex;

               if ((OperData->CellOffset + OperData->CellIndex)
                                                                  < (OperD
ata->Periods.PeriodCount - 1))
                  {
                                             OperData->CellOffset++;
                                             if (OperData->CellOffset >= Op
erData->CellCount)
                     {
                     OperData->CellOffset--;
                                                         OperData->CellIndex++;
                     }
                  }

                                    if (OperData->CellIndex == CellTempInd
ex)
                  MarkSelection(hWnd, hDC, OperData->Mark);
               else InvalidateRect(hWnd, NULL, TRUE);
               }
            else
               {
               GetTextMetrics(hDC, &tm);
               CellTempOffset = HIWORD(lParam) / (tm.tmHeight + 6);

                                    if ((OperData->CellIndex + CellTempOff
set)
                                                                  <= (Oper
Data->Parameters.YearCurrent
                              - OperData->Parameters.YearOpened))
                  {
                                             OperData->CellOffset = CellTem
pOffset;
                  }

               MarkSelection(hWnd, hDC, OperData->Mark);
                                    }
```



```
            ReleaseDC(hWnd, hDC);

            ret = TRUE;
            }
         else ret = FALSE;

         break;

               case WM_LBUTTONUP:
         if (OperData == NULL)
            {
                              ret = TRUE;
            break;
            }

                        hDC = GetDC(hWnd);

         GetTextMetrics(hDC, &tm);
         CellTempOffset = HIWORD(lParam) / (tm.tmHeight + 6);

         if ((OperData->CellIndex + CellTempOffset)
                  > (OperData->Parameters.YearCurrent
                        - OperData->Parameters.YearOpened))
            {
                              ReleaseDC(hWnd, hDC);
                              ReleaseCapture();
            ret = TRUE;
                              break;
                              }

         OperData->CellOffset = CellTempOffset;
         if ((OperData->CellOffset + OperData->CellIndex) == OperData->Mark)
                              OperData->Mark = 0;

         InvalidateRect(hWnd, NULL, TRUE);

         ReleaseDC(hWnd, hDC);

         ReleaseCapture();

         ret = TRUE;
                        break;

               case WM_PAINT:
                        if (OperData == NULL)
                              {
                              ret = TRUE;
                              break;
                              }

                        hDC = BeginPaint(hWnd, &lpps);

                        GetTextMetrics(hDC, &tm);
                        GetClientRect(hWnd, &r);

                        WindowWidth = r.right - r.left;
                        WindowHeight = r.bottom - r.top;

                        for (i = 0 ; i < WindowHeight ; i += tm.tmHeight + 6)
                              {
                              MoveToEx(hDC, 0, i, NULL);
                              LineTo(hDC, WindowWidth, i);

                              SetTextAlign(hDC, TA_CENTER);
            sprintf(YearString, "%4d", OperData->Parameters.YearOpened
                              + OperData->CellIndex + j++);
```

```
            TextOut(hDC, tm.tmMaxCharWidth * 2, i + 3, YearString, 4);

            UpdateEntry(hWnd, hDC, j - 1);

            MoveToEx(hDC, tm.tmMaxCharWidth * 4, i, NULL);
            LineTo(hDC, tm.tmMaxCharWidth * 4, i + tm.tmHeight + 6);

            MoveToEx(hDC, tm.tmMaxCharWidth * 17, i, NULL);
                        LineTo(hDC, tm.tmMaxCharWidth * 17, i + tm.tmH
eight + 6);


            if ((OperData->CellIndex + j) >= OperData->Periods.PeriodCount)
               {
               MoveToEx(hDC, 0, i + tm.tmHeight + 6, NULL);
               LineTo(hDC, WindowWidth, i + tm.tmHeight + 6);

               MoveToEx(hDC, tm.tmMaxCharWidth * 4, i, NULL);
               LineTo(hDC, tm.tmMaxCharWidth * 4, i + tm.tmHeight + 6);

               MoveToEx(hDC, tm.tmMaxCharWidth * 17, i, NULL);

               LineTo(hDC, tm.tmMaxCharWidth * 17, i + tm.tmHeight + 6);

               break;
               }
            }

         OperData->CellCount = WindowHeight / (tm.tmHeight + 4) - 1;

         if (OperData->Periods.PeriodCount <= OperData->CellCount)
           SetScrollRange(hWnd, SB_VERT, 0, 0, TRUE);
         else
           {
           SetScrollRange(hWnd, SB_VERT, 0, OperData->Periods.PeriodCount
                                - OperData->CellCount, FALSE);
           SetScrollPos(hWnd, SB_VERT, OperData->CellIndex, TRUE);
           }

         ret = SendMessage(RetrieveClientWindow(), WM_MDIGETACTIVE, 0, 0);

         if ((LOWORD(ret) == GetParent(hWnd)) || (HIWORD(ret)))
             HighlightCell(hWnd, hDC, OperData->CellOffset);

                        if (((OperData->CellIndex + OperData->CellOffset) != O
perData->Mark)
                                    && (OperData->Mark > 0))
            MarkSelection(hWnd, hDC, OperData->Mark);

         EndPaint(hWnd, &lpps);

         ret = TRUE;
         break;

      case AM_MDISAVESTATE:
         if (OperData == NULL)
            {
            ret = TRUE;
            break;
            }

         hDC = GetDC(hWnd);
                     UnHighlightCell(hWnd, hDC, OperData->CellOffset);
         ReleaseDC(hWnd, hDC);
         ret = TRUE;
         break;
```



```
      case AM_MDIRESTORESTATE:
         if (OperData == NULL)
            {
            ret = TRUE;
            break;
            }

                        hDC = GetDC(hWnd);
         HighlightCell(hWnd, hDC, OperData->CellOffset);

         InvalidateRect(hWnd, NULL, TRUE);

                        ReleaseDC(hWnd, hDC);
         ret = TRUE;
         break;

      case AM_PASTE:
         if (OperData == NULL)
            {
            ret = TRUE;
            break;
            }

         if (IsClipboardFormatAvailable(CF_TEXT))
            {
            if (OpenClipboard(hWnd))
               {
               hData = GetClipboardData(CF_TEXT);

                                    if (hData != NULL)
                  {
                  Text = (char *) GlobalLock(hData);
                  Period = (double *) GlobalLock(OperData->Periods.hPeriod);

                  if (Period != NULL)
                     {
                     for (i = OperData->CellIndex + OperData->CellOffset ;
                          i < OperData->CellCount ; i++)

                        Period[i] = strtod(Text, &Text);

                        if ((*Text == '\0') || (*(Text + 1) == '\0')) break;
                        Text++;
                        }

                     for (i = 0 ; i < OperData->Periods.PeriodCount ; i++)

                        if (i == 0) Period[i] = 0.0;
                        else
                           {
                           if (Period[i] < Period[i - 1])
                              Period[i] = Period[i - 1];

                           if (Period[i] > OperData->Parameters.Capacity)
                              Period[i] = OperData->Parameters.Capacity;
                           }
                        }
                     }

                  GlobalUnlock(OperData->Periods.hPeriod);
                  GlobalUnlock(hData);

                  InvalidateRect(hWnd, NULL, TRUE);
                                       }
```

```
                CloseClipboard();
                                    }
            }

         ret = TRUE;
         break;

      case AM_CUT:
         if (OperData == NULL)
            {
            ret = TRUE;
            break;
            }

         if (OpenClipboard(hWnd))
                                 {
            EmptyClipboard();

            hCut = CutPeriods(hWnd, OperData->Mark);

            if (hCut != NULL)
               {
               SetClipboardData(CF_TEXT, hCut);
               CloseClipboard();

               InvalidateRect(hWnd, NULL, TRUE);
               }
                                 }

         ret = TRUE;
         break;

               default:
         ret = DefWindowProc(hWnd, iMessage, wParam, lParam);
         break;
      }

   if ((iMessage != WM_DESTROY) && (iMessage != WM_CREATE))
      if (GetWindowLong(hWnd, 0) !=  NULL)
         GlobalUnlock(GetWindowLong(hWnd, 0));

   return ret;
}



// Handles the selection and display of the current year


void HighlightCell(HWND hWnd, HDC hDC, int Cell)
{
HWND Tool;
OPERDATASTRUCT *OperData;
TEXTMETRIC tm;
HPEN hPen, hOldPen;
HBRUSH hBrush, hOldBrush;
LOGBRUSH lb;
RECT r;
int WindowWidth;

   OperData = (OPERDATASTRUCT *) GlobalLock(GetWindowLong(hWnd, 0));

   GetTextMetrics(hDC, &tm);

   GetClientRect(hWnd, &r);
```



```
   WindowWidth = r.right - r.left;

        hPen = CreatePen(PS_SOLID, 2, RGB(0, 0, 0));
   hOldPen = SelectObject(hDC, hPen);

   lb.lbStyle = BS_HOLLOW;
   hBrush = CreateBrushIndirect(&lb);
   hOldBrush = SelectObject(hDC, hBrush);

   Rectangle(hDC, tm.tmMaxCharWidth * 4 + 2,
                  Cell * (tm.tmHeight + 6) + 2,
                  WindowWidth,
                  Cell * (tm.tmHeight + 6) + (tm.tmHeight + 6));

   SelectObject(hDC, hOldPen);
   DeleteObject(hPen);
   SelectObject(hDC, hOldBrush);
   DeleteObject(hBrush);

        Tool = SendMessage(RetrieveFrameWindow(), AM_GETACTIVETOOL, 0, 0);

// Added 08/15/95 - RLB

   if ((OperData->CellIndex + OperData->CellOffset)
         >= OperData->Periods.PeriodCount)
      {
      UnHighlightCell(hWnd, hDC, OperData->CellIndex - 1);
      OperData->CellOffset--;
      if (OperData->CellOffset < 0)
         {
         OperData->CellOffset = 0;
         OperData->CellIndex
            = (OperData->CellIndex > 0) ? OperData->CellIndex-- : 0;
         }
      }

// End Mod 08/15/95

   SendMessage(GetWindow(Tool, GW_CHILD), AM_ENTRYSELECTED,
               OperData->CellIndex + OperData->CellOffset, 0);

   GlobalUnlock(GetWindowLong(hWnd, 0));
}



// Returns the cell from highlighted to normal state


void UnHighlightCell(HWND hWnd, HDC hDC, int Cell)
{
TEXTMETRIC tm;
HPEN hPen, hOldPen;
HBRUSH hBrush, hOldBrush;
LOGBRUSH lb;
RECT r;
int WindowWidth;

   GetTextMetrics(hDC, &tm);

   GetClientRect(hWnd, &r);

   WindowWidth = r.right - r.left;

   hPen = CreatePen(PS_SOLID, 2, RGB(255, 255, 255));
   hOldPen = SelectObject(hDC, hPen);
```

```
        lb.lbStyle = BS_HOLLOW;
    hBrush = CreateBrushIndirect(&lb);
    hOldBrush = SelectObject(hDC, hBrush);

    Rectangle(hDC, tm.tmMaxCharWidth * 4 + 2,
                                        Cell * (tm.tmHeight + 6) + 2,
                  WindowWidth,
                  Cell * (tm.tmHeight + 6) + (tm.tmHeight + 6));

    SelectObject(hDC, hOldPen);

    MoveToEx(hDC, tm.tmMaxCharWidth * 17, Cell * (tm.tmHeight + 6), NULL);

    LineTo(hDC, tm.tmMaxCharWidth * 17, (Cell + 1) * (tm.tmHeight + 6));

//    SelectObject(hDC, hOldPen);
    DeleteObject(hPen);
    SelectObject(hDC, hOldBrush);
    DeleteObject(hBrush);
]


// Updates a cell on the screen


void UpdateEntry(HWND hWnd, HDC hDC, int Cell)
{
HWND Tool;
TEXTMETRIC tm;
HPEN hPen, hOldPen;
HBRUSH hBrush, hOldBrush;
LOGBRUSH lb;
RECT r;
double *Period;
double AcceptRate;
OPERDATASTRUCT *OperData;
char RefuseString[40];
int WindowWidth;

    OperData = (OPERDATASTRUCT *) GlobalLock(GetWindowLong(hWnd, 0));

    if ((OperData->Periods.PeriodCount != 0) && (OperData->Periods.hPeriod != NUL
L))
        Period = (double *) GlobalLock(OperData->Periods.hPeriod);
    else return;

    GetTextMetrics(hDC, &tm);

    GetClientRect(hWnd, &r);
    WindowWidth = r.right - r.left;

    hPen = CreatePen(PS_NULL, 2, RGB(255, 255, 255));
    hOldPen = SelectObject(hDC, hPen);

    lb.lbStyle = BS_HOLLOW;
    lb.lbColor = RGB(255, 255, 255);
    hBrush = CreateBrushIndirect(&lb);
    hOldBrush = SelectObject(hDC, hBrush);

    Rectangle(hDC, tm.tmMaxCharWidth * 4 + 4,
                   Cell * (tm.tmHeight + 6) + 4,
                   WindowWidth - 2,
                   Cell * (tm.tmHeight + 6) + (tm.tmHeight + 6) - 2);

    SetTextAlign(hDC, TA_LEFT);
```



```
    if ((Cell + OperData->CellIndex) < (OperData->Periods.PeriodCount - 1))
        AcceptRate = Period[Cell + OperData->CellIndex + 1]
                   - Period[Cell + OperData->CellIndex];
    else AcceptRate = 0.0;

        sprintf(RefuseString, "%E", AcceptRate);
    TextOut(hDC, tm.tmMaxCharWidth * 7, Cell * (tm.tmHeight + 6) + 3, RefuseStrin
g,
            strlen(RefuseString));

        sprintf(RefuseString, "%E", Period[Cell + OperData->CellIndex]);
    TextOut(hDC, tm.tmMaxCharWidth * 20, Cell * (tm.tmHeight + 6) + 3, RefuseStri
ng,
            strlen(RefuseString));

    SelectObject(hDC, hOldPen);
    DeleteObject(hPen);
    SelectObject(hDC, hOldBrush);
    DeleteObject(hBrush);

    GlobalUnlock(OperData->Periods.hPeriod);
    GlobalUnlock(GetWindowLong(hWnd, 0));
]


// Marks the selection between the mouse pointer and where it began


void MarkSelection(HWND hWnd, HDC hDC, int Mark)
{
HWND Tool;
TEXTMETRIC tm;
HPEN hWhitePen, hBlackPen, hOldPen;
HBRUSH hWhiteBrush, hBlackBrush, hOldBrush;
LOGBRUSH lb;
RECT r;
int MaxDisplayed, MinDisplayed;
int MinSelect, MaxSelect;
int Index, Cell;
double *Period;
double AcceptRate;
OPERDATASTRUCT *OperData;
char RefuseString[40];
int WindowWidth;

    OperData = (OPERDATASTRUCT *) GlobalLock(GetWindowLong(hWnd, 0));

    if ((OperData->Periods.PeriodCount != 0) && (OperData->Periods.hPeriod != NUL
L))
        Period = (double *) GlobalLock(OperData->Periods.hPeriod);
    else return;

    GetTextMetrics(hDC, &tm);

    GetClientRect(hWnd, &r);
    WindowWidth = r.right - r.left;

    hBlackPen = CreatePen(PS_NULL, 2, RGB(0, 0, 0));

    lb.lbStyle = BS_SOLID;
    lb.lbColor = RGB(0, 0, 0);
    hBlackBrush = CreateBrushIndirect(&lb);

    hOldPen = SelectObject(hDC, hBlackPen);
    hOldBrush = SelectObject(hDC, hBlackBrush);
```

```
   MinSelect = Mark < (OperData->CellOffset + OperData->CellIndex) ?
               Mark : (OperData->CellOffset + OperData->CellIndex);
   MaxSelect = Mark > (OperData->CellOffset + OperData->CellIndex) ?

               Mark : (OperData->CellOffset + OperData->CellIndex);

   for (Cell = 0 ; Cell < OperData->CellCount ; Cell++)
      {
      Index = Cell + OperData->CellIndex;

      if ((Index >= MinSelect) && (Index <= MaxSelect))
         {
         Rectangle(hDC, tm.tmMaxCharWidth * 4 + 2,
                  Cell * (tm.tmHeight + 6) + 2,
                  WindowWidth,
                  Cell * (tm.tmHeight + 6) + (tm.tmHeight + 6));

         SetBkColor(hDC, RGB(0, 0, 0));
         SetTextColor(hDC, RGB(255, 255, 255));
         SetTextAlign(hDC, TA_LEFT);

         if ((Cell + OperData->CellIndex) > 0)
            AcceptRate = Period[Cell + OperData->CellIndex]
                       - Period[Cell + OperData->CellIndex - 1];
         else AcceptRate = 0.0;

        sprintf(RefuseString, "%E", AcceptRate);
         TextOut(hDC, tm.tmMaxCharWidth * 7, Cell * (tm.tmHeight + 6) + 3, Refus
eString,
                  strlen(RefuseString));

        sprintf(RefuseString, "%E", Period[Cell + OperData->CellIndex]);
         TextOut(hDC, tm.tmMaxCharWidth * 20, Cell * (tm.tmHeight + 6) + 3, Refu
seString,
                  strlen(RefuseString));
         }
      }

   SelectObject(hDC, hOldPen);
   DeleteObject(hBlackPen);
   SelectObject(hDC, hOldBrush);
   DeleteObject(hBlackBrush);

   GlobalUnlock(OperData->Periods.hPeriod);
   GlobalUnlock(GetWindowLong(hWnd, 0));
}



HGLOBAL CutPeriods(HWND hWnd, int Mark)
{
HGLOBAL hCut;
int MinSelect, MaxSelect;
OPERDATASTRUCT *OperData;
double *Period;
char *TextCut;
int i;

   OperData = (OPERDATASTRUCT *) GlobalLock(GetWindowLong(hWnd, 0));

   if ((OperData->Periods.PeriodCount != 0) && (OperData->Periods.hPeriod != NUL
L))
      Period = (double *) GlobalLock(OperData->Periods.hPeriod);
   else return NULL;
```



```
   MinSelect = Mark < (OperData->CellOffset + OperData->CellIndex) ?
               Mark : (OperData->CellOffset + OperData->CellIndex);
   MaxSelect = Mark > (OperData->CellOffset + OperData->CellIndex) ?

               Mark : (OperData->CellOffset + OperData->CellIndex);


   if ((MaxSelect - MinSelect + 1) == 0)
      {
      GlobalUnlock(OperData->Periods.hPeriod);
      GlobalUnlock(GetWindowLong(hWnd, 0));
      return NULL;
      }

   hCut = GlobalAlloc(GHND, (MaxSelect - MinSelect + 1) * 15);
   TextCut = (char *) GlobalLock(hCut);

   for (i = MinSelect ; i <= MaxSelect ; i++)
      {
      TextCut += sprintf(TextCut, "%E\n", Period[i]);

      Period[i] = 0;
      }

   for (i = 1 ; i < OperData->Periods.PeriodCount ; i++)
      if (Period[i] < Period[i - 1]) Period[i] = Period[i - 1];

   GlobalUnlock(hCut);
   GlobalUnlock(OperData->Periods.hPeriod);
   GlobalUnlock(GetWindowLong(hWnd, 0));

   return hCut;
}



void ScrollUp(HWND hWnd)
{
OPERDATASTRUCT *OperData;

   OperData = (OPERDATASTRUCT *) GlobalLock(GetWindowLong(hWnd, 0));

   if (OperData->CellIndex == 0) return;

   OperData->CellIndex--;

   SetScrollPos(hWnd, SB_VERT, OperData->CellIndex, TRUE);

   InvalidateRect(hWnd, NULL, TRUE);

   GlobalUnlock(GetWindowLong(hWnd, 0));
}


void ScrollDown(HWND hWnd)
{
OPERDATASTRUCT *OperData;

   OperData = (OPERDATASTRUCT *) GlobalLock(GetWindowLong(hWnd, 0));

   if ((OperData->Periods.PeriodCount - 1)
            <= (OperData->CellIndex + OperData->CellOffset)) return;

   OperData->CellIndex++;
```

```
   SetScrollPos(hWnd, SB_VERT, OperData->CellIndex, TRUE);

   InvalidateRect(hWnd, NULL, TRUE);

   GlobalUnlock(GetWindowLong(hWnd, 0));
}



void PageUp(HWND hWnd)
{
OPERDATASTRUCT *OperData;

   OperData = (OPERDATASTRUCT *) GlobalLock(GetWindowLong(hWnd, 0));

   if (OperData->CellIndex == 0) return;

   OperData->CellIndex -= OperData->CellCount;
   if (OperData->CellIndex < 0) OperData->CellIndex = 0;

   SetScrollPos(hWnd, SB_VERT, OperData->CellIndex, TRUE);

   InvalidateRect(hWnd, NULL, TRUE);

   GlobalUnlock(GetWindowLong(hWnd, 0));
}



void PageDown(HWND hWnd)
{
OPERDATASTRUCT *OperData;
int MinScroll, MaxScroll;

   OperData = (OPERDATASTRUCT *) GlobalLock(GetWindowLong(hWnd, 0));

   if ((OperData->CellIndex + OperData->CellOffset)
               >= OperData->Periods.PeriodCount) return;

   GetScrollRange(hWnd, SB_VERT, &MinScroll, &MaxScroll);
   if ((MaxScroll - MinScroll) == 0) return;

   OperData->CellIndex += OperData->CellCount;
   if ((OperData->CellIndex + OperData->CellOffset)
               >= OperData->Periods.PeriodCount)
      {
      OperData->CellIndex = OperData->Periods.PeriodCount - 1;
      OperData->CellOffset = 0;
      }

   SetScrollPos(hWnd, SB_VERT, OperData->CellIndex, TRUE);

   InvalidateRect(hWnd, NULL, TRUE);

   GlobalUnlock(GetWindowLong(hWnd, 0));
}



void MemAllocError(void)
{
   MessageBox(NULL, "Unable to Allocate Data Memory", "Memory Alloc Failed",
               MB_ICONSTOP | MB_OK);
}
```



```
BOOL AllocOperMemory(HWND hWnd)
{
HGLOBAL hOperData, hTemp;
OPERDATASTRUCT *OperData;
double *DestPeriod, *SourcePeriod;
AIRTOXIC *DestAirToxic, *SourceAirToxic;
void MemAllocError(void);

/* Allocate the memory to hold the specifics of the window */

   hOperData = GlobalAlloc(GHND, sizeof(OPERDATASTRUCT));
   if (hOperData == NULL)
      {
      MemAllocError();
      return FALSE;
      }

   OperData = (OPERDATASTRUCT *) GlobalLock(hOperData);

   OperData->AirToxics.AirToxicCount = 0;
   OperData->AirToxics.hAirToxic = NULL;

   OperData->Periods.PeriodCount = 0;
   OperData->Periods.hPeriod = NULL;

   OperData->CellIndex  = 0;
   OperData->CellOffset = 0;
   OperData->CellCount  = 0;

   GlobalUnlock(hOperData);

   SetWindowLong(hWnd, 0, (long) hOperData);

   return TRUE;
}



void FreeOperMemory(HWND hWnd)
{
HGLOBAL hOperData;
OPERDATASTRUCT *OperData;

   hOperData = (HGLOBAL) GetWindowLong(hWnd, 0);
   OperData = (OPERDATASTRUCT *) GlobalLock(hOperData);

   if (OperData->Periods.PeriodCount != 0)
      GlobalFree(OperData->Periods.hPeriod);

   if (OperData->AirToxics.AirToxicCount != 0)
      GlobalFree(OperData->AirToxics.hAirToxic);

   GlobalUnlock(hOperData);
   GlobalFree(hOperData);
}
```

```
// The following functions have to do with determining the handles
// of other windows that are derived from the OPERDATASTRUCT and to
// coordinate their actions.

void DestroyAllRelations(HWND hOperWind)
{
HWND WindowPtr;
HGLOBAL hOperData, hOperTest;

   hOperData = MDIGETOPERDATA(hOperWind);

   for (WindowPtr = GetWindow(hOperWind, GW_HWNDFIRST);
        WindowPtr != NULL ;
        WindowPtr = GetWindow(WindowPtr, GW_HWNDNEXT))
      {
      hOperTest = MDIGETOPERDATA(WindowPtr);

      if (hOperTest == hOperData) PostMessage(WindowPtr, WM_CLOSE, 0, 0);
      }
}




void InformAllRelations(HWND hOperWind, UINT Message, WPARAM wParam,
                        LPARAM lParam)
{
HWND WindowPtr;
HGLOBAL hOperData, hOperTest;

   hOperData = MDIGETOPERDATA(hOperWind);

   for (WindowPtr = GetWindow(hOperWind, GW_HWNDFIRST);
        WindowPtr != NULL ;
        WindowPtr = GetWindow(WindowPtr, GW_HWNDNEXT))
      {
      hOperTest = MDIGETOPERDATA(WindowPtr);

      if ((hOperTest == hOperData) && (hOperWind != WindowPtr))
         PostMessage(WindowPtr, Message, wParam, lParam);
      }
}
```

```
// ============================================================================
//                                OPERTOOL.C
// ============================================================================
// ============================================================================
//         The toolbar used to edit operational parameters data
// ----------------------------------------------------------------------------
// ----------------------------------------------------------------------------
//                             Revision History
// ----------------------------------------------------------------------------
//  09/30/94    Radian(RLB)              Initial Revision
// ============================================================================

// The most important thing to remember in this routine is the fact that
// it is just a plain dialog box.  It synchronizes data with external
// windows via custom messages.

#include <windows.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <mem.h>
#include <dir.h>
#include <ctype.h>

#include "landwin.h"
#include "landoper.h"
#include "str_ops.h"


#define UP_3D    1
#define DOWN_3D  2


#pragma argsused

BOOL CALLBACK OperDlgProc(HWND hWndDlg, unsigned uiMsg, WORD wParam,
                          LONG lParam)
{
int entry, state, count;
double *Period, value;
HDC hDC;
HGDIOBJ brush;
HWND ActiveWindow;
HGLOBAL hOperData;
OPERDATASTRUCT *OperData;
BOOL ValuesChanged, ret;
MSG Message;
char string[40];
void HandleYearOpenChg(HWND hWndDlg, OPERDATASTRUCT *OperData);
void HandleYearCurrentChg(HWND hWndDlg, OPERDATASTRUCT *OperData);
extern int WhichMDIType(HWND hWnd);
LRESULT CALLBACK OperIntNumberEditProc(HWND hWnd, UINT iMessage, WPARAM wParam,
                                       LPARAM lParam);
LRESULT CALLBACK OperFPNumberEditProc(HWND hWnd, UINT iMessage, WPARAM wParam,
                                      LPARAM lParam);


   switch (uiMsg)
      {
      case WM_INITDIALOG:
         CustomizeControl(hWndDlg, IDC_OPENYEAR, (FARPROC) OperIntNumberEditProc
);
         CustomizeControl(hWndDlg, IDC_CURRENTYEAR, (FARPROC) OperIntNumberEditP
roc);
         CustomizeControl(hWndDlg, IDC_CAPACITY, (FARPROC) OperFPNumberEditProc)
;
         CustomizeControl(hWndDlg, IDC_VALUEEDIT, (FARPROC) OperFPNumberEditProc
```



```
);
         CheckDlgButton(hWndDlg, IDC_RADIOBUTTON1, 1);
                        return FALSE;

      case WM_PAINT:
         ActiveWindow = SendMessage(RetrieveClientWindow(), WM_MDIGETACTIVE,
                                    0, 0);

         if (ActiveWindow != NULL)
            {
            if (WhichMDIType(ActiveWindow) != OPERWINDOW)
               return FALSE;

            hOperData = SendMessage(ActiveWindow, AM_GETOPERHANDLE, 0, 0);
                                    if (hOperData != NULL)
                                          OperData = (OPERDATASTRUCT *) GlobalLock
(hOperData);
                                    else return FALSE;
            }
         else return FALSE;

         sprintf(string, "%4d", OperData->Parameters.YearOpened);
         SetDlgItemText(hWndDlg, IDC_OPENYEAR, string);

                        sprintf(string, "%4d", OperData->Parameters.YearCurrent)
;
                        SetDlgItemText(hWndDlg, IDC_CURRENTYEAR, string);

                        sprintf(string, "%E", OperData->Parameters.Capacity);
                        SetDlgItemText(hWndDlg, IDC_CAPACITY, string);

/*          if (OperData->Parameters.Capacity > 0)
                                    EnableWindow(GetDlgItem(hWndDlg, IDC_VALUEEDIT),
 TRUE);
         else
            EnableWindow(GetDlgItem(hWndDlg, IDC_VALUEEDIT), FALSE);
*/
         entry = (int) SendMessage(ActiveWindow, AM_GETCURRENTENTRY, 0, 0);

         Period = (double *) GlobalLock(OperData->Periods.hPeriod);

         if (IsDlgButtonChecked(hWndDlg, IDC_RADIOBUTTON1))
                                    sprintf(string, "%E", Period[entry]);
         else
            {
            if (entry == 0)
               sprintf(string, "%E", Period[entry]);
            else
                                          sprintf(string, "%E", Period[entry] - Pe
riod[entry - 1]);
            }

                        GlobalUnlock(OperData->Periods.hPeriod);
                        GlobalUnlock(hOperData);

         SetDlgItemText(hWndDlg, IDC_VALUEEDIT, string);
         return FALSE;

       case WM_CTLCOLOR:
         if (HIWORD(lParam) == CTLCOLOR_DLG) return GetStockObject(LTGRAY_BRUSH)
;
         else return NULL;

       case WM_SETFOCUS:
         SetFocus(GetDlgItem(hWndDlg, IDC_OPENYEAR));
         return TRUE;
```

```
                case AM_ENTRYSELECTED:
// The opertool toolbar actually absorbs a great deal of the overhead
// associated with the editing of the study.  In addition to changing
// individual values, this routine will also scan and correct values
// within the actual operating parameter data.

                        ActiveWindow = SendMessage(RetrieveClientWindow(), WM_MD
IGETACTIVE,

                0, 0);

                        if (ActiveWindow != NULL)
                                {
                                hOperData = SendMessage(ActiveWindow, AM_GETOPER
HANDLE, 0, 0);

                                if (hOperData != NULL)
                                        OperData = (OPERDATASTRUCT *) GlobalLock
(hOperData);
                                else return FALSE;
            }
         else return FALSE;

                        if (OperData->Periods.hPeriod != NULL)
            {
            Period = (double *) GlobalLock(OperData->Periods.hPeriod);

            entry = (int) wParam;

            count = OperData->Parameters.YearCurrent
                        - OperData->Parameters.YearOpened;

            if (entry < 0)
               {
               GlobalUnlock(OperData->Periods.hPeriod);
               GlobalUnlock(hOperData);
               return TRUE;
               }

            if (IsDlgButtonChecked(hWndDlg, IDC_RADIOBUTTON1))
               {
               if (entry == 0)
                  sprintf(string, "%E", 0.0);
               else
                  sprintf(string, "%E", Period[entry]);
                                }

            else
               {
                                if (entry == 0)
                  sprintf(string, "%E", Period[entry + 1]);
               else if (entry == count)
                  sprintf(string, "%E", 0.0);
               else
                                        sprintf(string, "%E", Period[ent
ry + 1] - Period[entry]);
               }

            GlobalUnlock(OperData->Periods.hPeriod);
            }

         GlobalUnlock(hOperData);

         SetDlgItemText(hWndDlg, IDC_VALUEEDIT, string);
         return TRUE;
```



```
      case WM_COMMAND:
        if ((int) wParam == IDC_RADIOBUTTON1)
            {
                        ActiveWindow = SendMessage(RetrieveClientWindow(), WM_MD
IGETACTIVE,

                0, 0);

            if (ActiveWindow != NULL)
                                {
                                hOperData = SendMessage(ActiveWindow, AM_GETOPER
HANDLE, 0, 0);

                                   if (hOperData != NULL)
                                        OperData = (OPERDATASTRUCT *) GlobalLock
(hOperData);
                                else return FALSE;
                                }
            else return FALSE;

            Period = (double *) GlobalLock(OperData->Periods.hPeriod);

            entry = (int) SendMessage(ActiveWindow, AM_GETCURRENTENTRY, 0, 0);

            if (entry == 0)
               sprintf(string, "%E", 0.0);
            else
               sprintf(string, "%E", Period[entry]);

            GlobalUnlock(OperData->Periods.hPeriod);
            GlobalUnlock(hOperData);

            SetDlgItemText(hWndDlg, IDC_VALUEEDIT, string);

            return FALSE;
            }
         else if ((int) wParam == IDC_RADIOBUTTON2)
            {
                        ActiveWindow = SendMessage(RetrieveClientWindow(), WM_MD
IGETACTIVE,

                0, 0);

            if (ActiveWindow != NULL)
                                {
                                hOperData = SendMessage(ActiveWindow, AM_GETOPER
HANDLE, 0, 0);

                                if (hOperData != NULL)
                                        OperData = (OPERDATASTRUCT *) GlobalLock
(hOperData);
                                else return FALSE;
                                }
            else return FALSE;

            Period = (double *) GlobalLock(OperData->Periods.hPeriod);

            entry = (int) SendMessage(ActiveWindow, AM_GETCURRENTENTRY, 0, 0);

                                if (entry == 0)
               sprintf(string, "%E", Period[entry + 1]);
            else if (entry == (OperData->Periods.PeriodCount - 1))
               sprintf(string, "%E", 0.0);
            else
                                        sprintf(string, "%E", Period[entry + 1]
- Period[entry]);
```

```
            GlobalUnlock(OperData->Periods.hPeriod);
            GlobalUnlock(hOperData);

            SetDlgItemText(hWndDlg, IDC_VALUEEDIT, string);

            return FALSE;
            }
         else if (HIWORD(lParam) != EN_KILLFOCUS) return FALSE;

       case AM_CONFIRMDATA:
         ActiveWindow = SendMessage(RetrieveClientWindow(), WM_MDIGETACTIVE,

                0, 0);

         if (ActiveWindow != NULL)
                                  {
                                  hOperData = SendMessage(ActiveWindow, AM_GETOPER
HANDLE, 0, 0);

                                  if (hOperData != NULL)
                                         OperData = (OPERDATASTRUCT *) GlobalLock
(hOperData);
                                  else return FALSE;
                                  }
         else return FALSE;

         switch(wParam)
            {
            case IDC_OPENYEAR:
               HandleYearOpenChg(hWndDlg, OperData);
               SendMessage(ActiveWindow, AM_VALUESCHANGED, -1, 0);
               break;

            case IDC_CURRENTYEAR:
               HandleYearCurrentChg(hWndDlg, OperData);
               SendMessage(ActiveWindow, AM_VALUESCHANGED, -1, 0);
               break;

            case IDC_CAPACITY:
               GetDlgItemText(hWndDlg, IDC_CAPACITY, string, 40);
               OperData->Parameters.Capacity = strtod(string, NULL);

               count = OperData->Parameters.YearCurrent
                     - OperData->Parameters.YearOpened + 1;
               ValuesChanged = FALSE;

               Period = (double *) GlobalLock(OperData->Periods.hPeriod);

               for (entry = 0 ; entry < count ; entry++)
                  if (Period[entry] > OperData->Parameters.Capacity)
                     {
                     if (entry > 0) Period[entry] = Period[entry - 1];
                     else Period[entry] = 0;

                     ValuesChanged = TRUE;
                                                   }

               SendMessage(ActiveWindow, AM_VALUESCHANGED, -1, 0);

               GlobalUnlock(OperData->Periods.hPeriod);

               sprintf(string, "%E", OperData->Parameters.Capacity);
               SetDlgItemText(hWndDlg, IDC_CAPACITY, string);
               break;
```



```
            case IDC_VALUEEDIT:
               if (OperData->Periods.hPeriod != NULL)
                  {
                  Period = (double *) GlobalLock(OperData->Periods.hPeriod);

                  entry = (int) SendMessage(ActiveWindow, AM_GETCURRENTENTRY, 0,
0);

                  GetDlgItemText(hWndDlg, IDC_VALUEEDIT, string, 40);

                  ValuesChanged = FALSE;

                  value = strtod(string, NULL);

                  if (value > OperData->Parameters.Capacity)
                     {
                     MessageBox(NULL, "Refuse in place cannot exceed capacity",
                                 "Validation Error", MB_OK | MB_ICONSTOP);
                     GlobalUnlock(OperData->Periods.hPeriod);

                     break;
                     }

                  count = OperData->Parameters.YearCurrent
                        - OperData->Parameters.YearOpened + 1;

                  if (entry == 0)
                     {
                     if (IsDlgButtonChecked(hWndDlg, IDC_RADIOBUTTON1))
                        {
                        if (value > 0)
                           {
                                                                        MessageB
ox(NULL, "Refuse in place cannot be greater "\
                                                "than zero during the first year",
                                                "Validation Error", MB_OK | MB_ICONS
TOP);
                           GlobalUnlock(OperData->Periods.hPeriod);

                           sprintf(string, "%E", 0.0);
                           SetDlgItemText(hWndDlg, IDC_VALUEEDIT, string);
                           break;
                           }
                        }
                     else if (Period[entry + 1] != value)
                        {
                        ValuesChanged = TRUE;
                        Period[entry + 1] = value;
                        }
                     }
                  else if (entry == (count - 1))
                     {
                     if (IsDlgButtonChecked(hWndDlg, IDC_RADIOBUTTON2))
                        {
                        if (value > 0)
                           {
                           MessageBox(NULL, "Acceptance rate cannot be greater "
\
                                                "than zero during the last year",

                     "Validation Error", MB_OK | MB_ICONSTOP);
                           GlobalUnlock(OperData->Periods.hPeriod);

                           sprintf(string, "%E", 0.0);
                           SetDlgItemText(hWndDlg, IDC_VALUEEDIT, string);
                           break;
```

```
                        }
                     }
                  else if (Period[entry] != value)
                     {
                     ValuesChanged = TRUE;
                     Period[entry] = value;
                     }
                  }
               else
                  {
                  if (IsDlgButtonChecked(hWndDlg, IDC_RADIOBUTTON1))
                     {
                     if (Period[entry - 1] > value)
                        {
                        MessageBox(NULL, "Refuse in place cannot decrease "\
                                         "over time.",
                                         "Validation Error", MB_OK | MB_ICONS
TOP);
                                                                    GlobalUn
lock(OperData->Periods.hPeriod);

                        break;
                        }

                     if (Period[entry] != value)
                        {
                        ValuesChanged = TRUE;
                        Period[entry] = value;
                        }
                     }
                  else
                     {
                     if ((Period[entry] + value) != Period[entry + 1])
                        {
                        ValuesChanged = TRUE;
                        Period[entry + 1] = Period[entry] + value;
                        }
                     }
                  }

// Insure the entries are self consistent.  If not, correct.

               for (entry = 1; entry < count; entry++)
                  {
                  if (Period[entry] < Period[entry - 1])
                                                        {
                     Period[entry] = Period[entry - 1];
                     ValuesChanged = TRUE;
                     }

/* Added 10/17/95 */

                  if (Period[entry] > OperData->Parameters.Capacity)
                     {
                     Period[entry] = OperData->Parameters.Capacity;
                     ValuesChanged = TRUE;
                     }
                  }

               if (ValuesChanged)
                  SendMessage(ActiveWindow, AM_VALUESCHANGED, -1, 0);

               GlobalUnlock(OperData->Periods.hPeriod);
               }

            break;
```



```
            default:
               return FALSE;
            }

         return TRUE;

      case AM_FLUSHFIELDS:
         ActiveWindow = (HWND) lParam;

         entry = (int) SendMessage(ActiveWindow, AM_GETCURRENTENTRY, 0, 0);
         if (entry < 0) return TRUE;

         hOperData = (HGLOBAL) SendMessage(ActiveWindow, AM_GETOPERHANDLE,

                                  0, 0);
                        if (hOperData != NULL)
                              OperData = (OPERDATASTRUCT *) GlobalLock(hOperDa
ta);
                        else return TRUE;

         HandleYearOpenChg(hWndDlg, OperData);
         HandleYearCurrentChg(hWndDlg, OperData);

         GetDlgItemText(hWndDlg, IDC_CAPACITY, string, 40);
         OperData->Parameters.Capacity = strtod(string, NULL);

         Period = (double *) GlobalLock(OperData->Periods.hPeriod);

         GetDlgItemText(hWndDlg, IDC_VALUEEDIT, string, 40);
         if (IsDlgButtonChecked(hWndDlg, IDC_RADIOBUTTON1))
            Period[entry] = strtod(string, NULL);
         else
            {
            Period[entry] = strtod(string, NULL);
            if (IsDlgButtonChecked(hWndDlg, IDC_RADIOBUTTON2) && (entry > 0))
               Period[entry] += Period[entry - 1];
            }

         count = OperData->Parameters.YearCurrent
                        - OperData->Parameters.YearOpened + 1;

         for (++entry; entry < count; entry++)
            if (Period[entry] < Period[entry - 1])
               Period[entry] = Period[entry - 1];

         SendMessage(ActiveWindow, AM_VALUESCHANGED, -1, 0);

         GlobalUnlock(OperData->Periods.hPeriod);
         GlobalUnlock(hOperData);
         return TRUE;

      case AM_UPDATEFIELDS:
         ActiveWindow = (HWND) lParam;
         if (ActiveWindow == NULL) return TRUE;

         entry = (int) SendMessage(ActiveWindow, AM_GETCURRENTENTRY, 0, 0);
         if (entry < 0) return TRUE;

         hOperData = (HGLOBAL) SendMessage(ActiveWindow, AM_GETOPERHANDLE,
                                           0, 0);
         if (hOperData != NULL)
            OperData = (OPERDATASTRUCT *) GlobalLock(hOperData);

                        else return TRUE;
```

```
                        sprintf(string, "%4d", OperData->Parameters.YearOpened);
                        SetDlgItemText(hWndDlg, IDC_OPENYEAR, string);

        sprintf(string, "%4d", OperData->Parameters.YearCurrent);
        SetDlgItemText(hWndDlg, IDC_CURRENTYEAR, string);

        sprintf(string, "%E", OperData->Parameters.Capacity);
        SetDlgItemText(hWndDlg, IDC_CAPACITY, string);

        if (OperData->Pericds.hPeriod != NULL)
           {
           Period = (double *) GlobalLock(OperData->Periods.hPeriod);

           if (IsDlgButtonChecked(hWndDlg, IDC_RADIOBUTTON1))
              sprintf(string, "%E", Period[entry]);
           else
              {
              if (entry == 0)
                 sprintf(string, "%E", Period[entry]);
                                  else
                 sprintf(string, "%E", Period[entry] - Period[entry - 1]);
              }

           GlobalUnlock(OperData->Periods.hPeriod);
           SetDlgItemText(hWndDlg, IDC_VALUEEDIT, string);
           }

        GlobalUnlock(OperData->Periods.hPeriod);
        GlobalUnlock(hOperData);
        return TRUE;

      default:
         return FALSE;
      }
}


// This function handles the change in the opening year by resizing
// the arrays that track the data


void HandleYearOpenChg(HWND hWndDlg, OPERDATASTRUCT *OperData)
{
int TempYear;
HGLOBAL hPeriod;
double *PeriodSrc, *PeriodDest;
char string[40];
HWND ActiveWindow;

   if (OperData == NULL) return;

   GetDlgItemText(hWndDlg, IDC_OPENYEAR, string, 40);

   if (atoi(string) == 0) return;
   if ((OperData->Parameters.YearCurrent - atoi(string)) < 0) return;

   TempYear = atoi(string);

   if ((OperData->Parameters.YearCurrent - TempYear + 1)
                > OperData->Periods.PeriodCount)
      {
      hPeriod = (HGLOBAL) GlobalAlloc(GHND,
               sizeof(double) * (OperData->Parameters.YearCurrent - TempYear + 1
));
      if (hPeriod == NULL) return;
```



```
      PeriodDest = (double *) GlobalLock(hPeriod);
      PeriodSrc = (double *) GlobalLock(OperData->Periods.hPeriod);

      memcpy((void *) (PeriodDest + (OperData->Parameters.YearOpened - TempYear)
),
             (void *) PeriodSrc,
             sizeof(double) * OperData->Periods.PeriodCount);

      GlobalUnlock(hPeriod);
      GlobalUnlock(OperData->Periods.hPeriod);

      GlobalFree(OperData->Periods.hPeriod);

      OperData->Periods.hPeriod = hPeriod;
      OperData->Periods.PeriodCount = OperData->Parameters.YearCurrent - TempYea
r + 1;

      OperData->Parameters.YearOpened = TempYear;

      ActiveWindow = SendMessage(RetrieveClientWindow(), WM_MDIGETACTIVE,
                                 0, 0);
      SendMessage(ActiveWindow, AM_PERIODCHANGED, 0, 0);

      }
   else if ((OperData->Parameters.YearCurrent - TempYear + 1)
             < OperData->Periods.PeriodCount)
      {
      hPeriod = (HGLOBAL) GlobalAlloc(GHND,
               sizeof(double) * (OperData->Parameters.YearCurrent - TempYear + 1
));
      if (hPeriod == NULL) return;

      PeriodDest = (double *) GlobalLock(hPeriod);
      PeriodSrc = (double *) GlobalLock(OperData->Periods.hPeriod);

      memcpy((void *) PeriodDest,
             (void *) (PeriodSrc + (TempYear - OperData->Parameters.YearOpened))
,
             sizeof(double) * (OperData->Parameters.YearCurrent - TempYear + 1))
;

      GlobalUnlock(hPeriod);
      GlobalUnlock(OperData->Periods.hPeriod);

      GlobalFree(OperData->Periods.hPeriod);

      OperData->Periods.hPeriod = hPeriod;
      OperData->Periods.PeriodCount = OperData->Parameters.YearCurrent - TempYea
r + 1;

      OperData->Parameters.YearOpened = TempYear;

      ActiveWindow = SendMessage(RetrieveClientWindow(), WM_MDIGETACTIVE,
                                 0, 0);
      SendMessage(ActiveWindow, AM_PERIODCHANGED, 0, 0);

      }
}



// This function handles the change in the current year by resizing
// the arrays that track the data

void HandleYearCurrentChg(HWND hWndDlg, OPERDATASTRUCT *OperData)
{
```

```
int TempYear;
HGLOBAL hPeriod;
double *PeriodSrc, *PeriodDest;
char string[40];
HWND ActiveWindow;

   if (OperData == NULL) return;

   GetDlgItemText(hWndDlg, IDC_CURRENTYEAR, string, 40);

   if (atoi(string) == 0) return;
   if ((atoi(string) - OperData->Parameters.YearOpened) < 0) return;

   TempYear = atoi(st...;

   if ((TempYear - OperData->Parameters.YearOpened + 1)
            > OperData->Periods.PeriodCount)
      {
      hPeriod = (HGLOBAL) GlobalAlloc(GHND,
            sizeof(double) * (TempYear - OperData->Parameters.YearOpened + 1)
);
      if (hPeriod == NULL) return;

      PeriodDest = (double *) GlobalLock(hPeriod);
      PeriodSrc = (double *) GlobalLock(OperData->Periods.hPeriod);

      memcpy((void *) PeriodDest,
             (void *) PeriodSrc,
             sizeof(double) * OperData->Periods.PeriodCount);

      GlobalUnlock(hPeriod);
      GlobalUnlock(OperData->Periods.hPeriod);

      GlobalFree(OperData->Periods.hPeriod);

      OperData->Periods.hPeriod = hPeriod;
      OperData->Periods.PeriodCount = TempYear - OperData->Parameters.YearOpened
 + 1;

      OperData->Parameters.YearCurrent = TempYear;

      ActiveWindow = SendMessage(RetrieveClientWindow(), WM_MDIGETACTIVE,
                           0, 0);
      SendMessage(ActiveWindow, AM_PERIODCHANGED, 0, 0);

      }
   else if ((TempYear - OperData->Parameters.YearOpened + 1)
            < OperData->Periods.PeriodCount)
      {
      hPeriod = (HGLOBAL) GlobalAlloc(GHND,
            sizeof(double) * (TempYear - OperData->Parameters.YearOpened + 1)
);
      if (hPeriod == NULL) return;

      PeriodDest = (double *) GlobalLock(hPeriod);
      PeriodSrc = (double *) GlobalLock(OperData->Periods.hPeriod);

      memcpy((void *) PeriodDest,
             (void *) PeriodSrc,
             sizeof(double) * (TempYear - OperData->Parameters.YearOpened + 1));

      GlobalUnlock(hPeriod);
      GlobalUnlock(OperData->Periods.hPeriod);

      GlobalFree(OperData->Periods.hPeriod);
```



```
      OperData->Periods.hPeriod = hPeriod;
      OperData->Periods.PeriodCount = TempYear - OperData->Parameters.YearOpened
 + 1;

      OperData->Parameters.YearCurrent = TempYear;

      ActiveWindow = SendMessage(RetrieveClientWindow(), WM_MDIGETACTIVE,
                           0, 0);
      SendMessage(ActiveWindow, AM_PERIODCHANGED, 0, 0);

      }
|



// Custom edit control procedures


LRESULT CALLBACK OperIntNumberEditProc(HWND hWnd, UINT iMessage, WPARAM wParam,
                                       LPARAM lParam)
{
HWND ActiveWindow;
LRESULT ret;
LRESULT CALLBACK CallSCProc(HWND hWnd, UINT iMessage, WORD wParam,
                            DWORD lParam);
LRESULT CALLBACK OperIntNumberEditProc(HWND hWnd, UINT iMessage, WPARAM wParam,
                                       LPARAM lParam);

   switch (iMessage)
      {
      case WM_GETDLGCODE:
         ret = CallSCProc(hWnd, iMessage, wParam, lParam);
         if (lParam)
            {
            if ((((MSG *) lParam)->message == WM_KEYDOWN) ||
                  (((MSG *) lParam)->message == WM_KEYUP) ||
                  (((MSG *) lParam)->message == WM_CHAR))
               if (((MSG *) lParam)->wParam == VK_RETURN)
                  ret |= DLGC_WANTMESSAGE;
            |
         return ret;

      case WM_NCDESTROY:
         UnCustomizeControl(hWnd);
         return TRUE;

      case WM_CHAR:
         if (wParam == VK_RETURN) return TRUE;

         if (isalnum((char) wParam))
            {
            if (isdigit((char) wParam))
               return CallSCProc(hWnd, iMessage, wParam, lParam);
            else return TRUE;
            }
         else return CallSCProc(hWnd, iMessage, wParam, lParam);

      case WM_PAINT:
         ret = CallSCProc(hWnd, iMessage, wParam, lParam);
         Draw3DFrame(hWnd, DOWN_3D);
         return ret;

       case WM_KEYDOWN:
         switch (wParam)
              |
              case VK_UP:
```

```
            case VK_DOWN:
            case VK_PRIOR:
            case VK_NEXT:
               ActiveWindow = SendMessage(RetrieveClientWindow(),
                                          WM_MDIGETACTIVE, 0, 0);
               SendMessage(ActiveWindow, iMessage, wParam, lParam);
               return TRUE;

            case VK_RETURN:
               SendMessage(GetParent(hWnd), AM_CONFIRMDATA,
                                GetDlgCtrlID(hWnd), 0);
               return 0;

            default:
               return CallSCProc(hWnd, iMessage, wParam, lParam);
            }


      case WM_KEYUP:
        switch (wParam)
            {
            case VK_UP:
            case VK_DOWN:
            case VK_PRIOR:
            case VK_NEXT:
               return TRUE;

            case VK_RETURN:
               return TRUE;

            default:
               return CallSCProc(hWnd, iMessage, wParam, lParam);
            }

     default:
        return CallSCProc(hWnd, iMessage, wParam, lParam);
     }
}


LRESULT CALLBACK OperFPNumberEditProc(HWND hWnd, UINT iMessage, WPARAM wParam,
                                      LPARAM lParam)
{
HWND ActiveWindow;
char string[30], oldstring[30];
LRESULT ret;
void UnCustomizeControl(HWND hWnd);
LRESULT CALLBACK CallSCProc(HWND hWnd, UINT iMessage, WORD wParam,
                            DWORD lParam);
NCCALCSIZE_PARAMS *lpncsp = (NCCALCSIZE_PARAMS *) lParam;
WINDOWPOS *WinPos;

   switch (iMessage)
      {
      case WM_GETDLGCODE:
         ret = CallSCProc(hWnd, iMessage, wParam, lParam);
         if (lParam)
            {
            if ((((MSG *) lParam)->message == WM_KEYDOWN) ||
                  (((MSG *) lParam)->message == WM_KEYUP) ||
                  (((MSG *) lParam)->message == WM_CHAR))
               if (((MSG *) lParam)->wParam == VK_RETURN)
                  ret |= DLGC_WANTMESSAGE;
            }
         return ret;
```



```
      case WM_NCDESTROY:
         UnCustomizeControl(hWnd);
         return TRUE;

      case WM_CHAR:
/*
         memset(string, 0, sizeof(string));
         SendMessage(hWnd, WM_GETTEXT, sizeof(string), (long) string);

         *(string + strlen(string)) = (char) wParam;

         if (isprint((char) wParam))
            {
            if (is_string_pos_number(string))
                 return CallSCProc(hWnd, iMessage, wParam, lParam);
            else return TRUE;
            }

         return CallSCProc(hWnd, iMessage, wParam, lParam);
*/
         if (wParam == VK_RETURN) return TRUE;

         memset(string, 0, sizeof(oldstring));
         SendMessage(hWnd, WM_GETTEXT, sizeof(oldstring), (LPARAM) oldstring);

         memset(string, 0, sizeof(string));
         ret = CallSCProc(hWnd, iMessage, wParam, lParam);

         SendMessage(hWnd, WM_GETTEXT, sizeof(string), (LPARAM) string);

         if (is_string_pos_number(string)) return ret;
         else SendMessage(hWnd, WM_SETTEXT, 0, (LPARAM) oldstring);

         return FALSE;

//       case WM_NCPAINT:
//          Draw3DFrame(hWnd, DOWN_3D);
//          UpdateWindow(hWnd);

//          return TRUE;

       case WM_NCCALCSIZE:
          if (! (BOOL) wParam) return FALSE;

          WinPos = lpncsp->lppos;

          CopyRect(((RECT *) lpncsp->rgrc) + 2, lpncsp->rgrc);

          lpncsp->rgrc[1].left += 2;
          lpncsp->rgrc[1].right -= 2;
          lpncsp->rgrc[1].top += 2;
          lpncsp->rgrc[1].bottom -= 2;

          lpncsp->rgrc[2].left += 2;
          lpncsp->rgrc[2].right -= 2;
          lpncsp->rgrc[2].top += 2;
          lpncsp->rgrc[2].bottom -= 2;

          CopyRect(lpncsp->rgrc, ((RECT *) lpncsp->rgrc) + 2);

          lpncsp->rgrc[0].left = WinPos->x + 2;
          lpncsp->rgrc[0].top = WinPos->y + 2;
          lpncsp->rgrc[0].right = WinPos->cx - 2;
          lpncsp->rgrc[0].bottom = WinPos->cy - 2;

          return WVR_VALIDRECTS & WVR_HREDRAW & WVR_VREDRAW;
```

```
      case WM_PAINT:
         CallSCProc(hWnd, iMessage, wParam, lParam);
         Draw3DFrame(hWnd, DOWN_3D);
         return TRUE;

       case WM_KEYDOWN:
         switch (wParam)
              {
              case VK_UP:
              case VK_DOWN:
              case VK_PRIOR:
              case VK_NEXT:
                 SendMessage(GetParent(hWnd), AM_CONFIRMDATA,
                                  GetDlgCtrlID(hWnd), 0);

                 ActiveWindow = SendMessage(RetrieveClientWindow(),
                                       WM_MDIGETACTIVE, 0, 0);
                 SendMessage(ActiveWindow, iMessage, wParam, lParam);
                 return FALSE;

              case VK_RETURN:
                 SendMessage(GetParent(hWnd), AM_CONFIRMDATA,
                                  GetDlgCtrlID(hWnd), 0);
//                    SetFocus(GetParent(hWnd));
                   SetFocus(hWnd);
                 return FALSE;

              default:
                 return CallSCProc(hWnd, iMessage, wParam, lParam);
              }

       case WM_KEYUP:
         switch (wParam)
              {
              case VK_UP:
              case VK_DOWN:
              case VK_PRIOR:
              case VK_NEXT:
                 return FALSE;

              case VK_RETURN:
                 return FALSE;

              default:
                 return CallSCProc(hWnd, iMessage, wParam, lParam);
              }

      default:
         return CallSCProc(hWnd, iMessage, wParam, lParam);
      }
}
```

```
// ============================================================================
//                                 OPTIONS.C
// ============================================================================
// ============================================================================
//                      Handles user entry of model options
// ============================================================================
// ============================================================================
//                            Revision History
// ============================================================================
//  09/30/94    Radian(RLB)              Initial Revision
// ============================================================================

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

#include "landwin.h"
#include "datadef.h"
#include "str_ops.h"
#include "helpdef.h"

BOOL CALLBACK DialogProc(HWND hWndDlg, unsigned uiMsg, WORD wParam,
                         LONG lParam);


BOOL GetOptions(HWND hWnd, HINSTANCE hInst)
{
FARPROC DlgProcInstance;
int ret_value;

   DlgProcInstance = MakeProcInstance(DialogProc, hInst);
   ret_value = DialogBox(hInst, "OPTIONS_DIALOG", hWnd, DlgProcInstance);
   FreeProcInstance(DlgProcInstance);

   return ret_value;
}


#pragma argsused

static BOOL CALLBACK DialogProc(HWND hWndDlg, unsigned uiMsg, WORD wParam,
                                LONG lParam)
{
char string[30];
int item;

   switch (uiMsg)
      {
      case WM_INITDIALOG:
         CenterWindow(hWndDlg);
         CustomizeControl(hWndDlg, IDC_VALUEEDIT, (FARPROC) IntNumberEditProc);

         if (Limits.DefaultsMode == REGULATORY) item = IDC_RADIOBUTTON1;
         else if (Limits.DefaultsMode == AP42)  item = IDC_RADIOBUTTON2;
         else if (Limits.DefaultsMode == USER)  item = IDC_RADIOBUTTON3;

         SendMessage(GetDlgItem(hWndDlg, item), BM_SETCHECK, TRUE, 0L);

         sprintf(string, "%3d", Limits.MaxYears);
         SetDlgItemText(hWndDlg, IDC_VALUEEDIT, string);
         return TRUE;

      case WM_COMMAND:
         switch (LOWORD(wParam))
            {
```



```
            case IDC_OK:
               if (SendMessage(GetDlgItem(hWndDlg, IDC_RADIOBUTTON1),
                     BM_GETCHECK, 0, 0L)) Limits.DefaultsMode = REGULATORY;
               else if (SendMessage(GetDlgItem(hWndDlg, IDC_RADIOBUTTON2),
                     BM_GETCHECK, 0, 0L)) Limits.DefaultsMode = AP42;
               else if (SendMessage(GetDlgItem(hWndDlg, IDC_RADIOBUTTON3),
                     BM_GETCHECK, 0, 0L)) Limits.DefaultsMode = USER;

               GetDlgItemText(hWndDlg, IDC_VALUEEDIT, string, sizeof(string));
               Limits.MaxYears = atoi(string);

               SaveModelLimits();

               EndDialog(hWndDlg, TRUE);
               return TRUE;

            case IDC_CANCEL:
               EndDialog(hWndDlg, FALSE);
               return TRUE;

            case IDC_HELP:
               WinHelp(hWndDlg, "LANDWIN.HLP", HELP_CONTEXT, FileOptions);
               return TRUE;

            case IDC_VALUEEDIT:
               return FALSE;

            default:
               break;
            }
      }

   return FALSE;
}
```

```
// ========================================================================
//                              RPRTSEL.C
// ========================================================================
// ========================================================================
//     Dialog for selecting the compound used in the output report
// ========================================================================
// ========================================================================
//                            Revision History
// ========================================================================
//  09/30/94    Radian(RLB)              Initial Revision
// ========================================================================

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

#include "landwin.h"
#include "str_ops.h"
#include "helpdef.h"

BOOL CALLBACK DialogProc(HWND hWndDlg, unsigned uiMsg, WORD wParam,
                         LONG lParam);


int GetSubstanceSelection(HWND hWnd, HINSTANCE hInst)
{
FARPROC DlgProcInstance;
int ret_value;

   DlgProcInstance = MakeProcInstance(DialogProc, hInst);
   ret_value = DialogBox(hInst, "REPORT_DIALOG", hWnd, DlgProcInstance);
   FreeProcInstance(DlgProcInstance);

   return ret_value;
}


#pragma argsused

static BOOL CALLBACK DialogProc(HWND hWndDlg, unsigned uiMsg, WORD wParam,
                                LONG lParam)
{
int i, ret;
HWND ActiveWindow;
HGLOBAL hOperData = NULL;
AIRTOXIC *AirToxic;
OPERDATASTRUCT *OperData;

   switch (uiMsg)
      {
      case WM_INITDIALOG:
         CenterWindow(hWndDlg);

         ActiveWindow = SendMessage(RetrieveClientWindow(), WM_MDIGETACTIVE,
                                    0, 0);
         if (ActiveWindow != NULL)
            hOperData = SendMessage(ActiveWindow, AM_GETOPERHANDLE, 0, 0);

         if (hOperData != NULL)
            OperData = (OPERDATASTRUCT *) GlobalLock(hOperData);

         SendDlgItemMessage(hWndDlg, IDC_CHEMLIST, LB_ADDSTRING, 0, (LPARAM) "Me
thane (CH4)");
         SendDlgItemMessage(hWndDlg, IDC_CHEMLIST, LB_ADDSTRING, 0, (LPARAM) "Ca
rbon Dioxide");
```



```
         SendDlgItemMessage(hWndDlg, IDC_CHEMLIST, LB_ADDSTRING, 0, (LPARAM) "NM
OC");

         if (OperData->AirToxics.AirToxicCount != 0)
            {
            AirToxic = (AIRTOXIC *) GlobalLock(OperData->AirToxics.hAirToxic);

            for (i = 0 ; i < OperData->AirToxics.AirToxicCount ; i++)
               SendDlgItemMessage(hWndDlg, IDC_CHEMLIST, LB_ADDSTRING, 0,
                                  (LPARAM) AirToxic[i].Name);

            GlobalUnlock(OperData->AirToxics.hAirToxic);
            }

         SendDlgItemMessage(hWndDlg, IDC_CHEMLIST, LB_SETCURSEL, 0, 0);

         if (hOperData != NULL) GlobalUnlock(hOperData);
         return TRUE;

      case WM_COMMAND:
         switch (LOWORD(wParam))
            {
            case IDC_OK:
               ret = SendDlgItemMessage(hWndDlg, IDC_CHEMLIST, LB_GETCURSEL,
                                        0, 0);
               EndDialog(hWndDlg, ret + 1);
               return TRUE;

            case IDC_CANCEL:
               EndDialog(hWndDlg, -1);
               return TRUE;

            case IDC_HELP:
               WinHelp(hWndDlg, "LANDWIN.HLP", HELP_CONTEXT, ReportSel);
               return TRUE;

            case IDC_VALUEEDIT:
               return FALSE;

            default:
               break;
            }
      }

   return FALSE;
}
```

```
// ==========================================================================
//                                STR_OPS.C
// ==========================================================================
// ==========================================================================
//                  Miscellaneous string handling routines
// ==========================================================================
// ==========================================================================
//                            Revision History
// ==========================================================================
//  09/30/94    Radian(RLB)             Initial Revision
// ==========================================================================

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

#include "str_ops.h"

#define FALSE  0
#define TRUE   1


char *find_noblank(string)
char *string;
{
   if (string == NULL) return(NULL);

   while (*string != '\0')
      {
      if (isgraph(*string)) break;
      string++;
      }

   return(string);
}



char *strip_spaces(string)
char *string;
{
char *ptr;

   if (*string == '\0') return(string);

   ptr = string + strlen(string) - 1;
   while ((! isgraph(*ptr)) && (*string != '\0')) *ptr-- = '\0';

   return(string);
}


char *strip_last_asterisk(string)
char *string;
{
char *ptr;

   if (*string == '\0') return(string);

   ptr = string + strlen(string) - 1;
   while ((((! isgraph(*ptr)) || (*ptr == '*')) && (ptr != string))
      *ptr-- = '\0';

   if ((! isgraph(*ptr)) || (*ptr == '*')) *ptr = '\0';
```



```
   return(string);
}


int remove_control(string)
char *string;
{
   while (*string != '\0')
      {
      if (*string < ' ') *string = ' ';
      ++string;
      }

   return(0);
}



char *strupper(string1, string2)
char *string1, *string2;
{
char *ptr;

   ptr = string1;
   while (*string2 != '\0')
      {
      *string1++ = (char) toupper((int) *string2);
      string2++;
      }

   *string1 = '\0';
   return(ptr);
}



char *strlower(string1, string2)
char *string1, *string2;
{
char *ptr;

   ptr = string1;
   while (*string1 != '\0') *string1++ = tolower(*string2++);

   *string1 = '\0';
   return(ptr);
}


/* ========================================================================= */
/* This function evaluates whether a given string is a valid base 10         */
/* integer or floating point number.  The valid format for the               */
/* number is                                                                 */
/*                                                                           */
/*      [sign][numbers][.][numbers][exponentiation][sign][numbers]           */
/*                                                                           */
/* ========================================================================= */

int is_string_number(string)
char *string;
{
/* First, determine if the beginning is a sign operator. */

   string = find_noblank(string);

   if ((*string == '+') || (*string == '-')) string++;
```

```
   while ((*string != '\0') && (*string != '.') &&
          ((char) toupper(*string) != 'E'))
      if (! isdigit((int) *string++)) return(FALSE);

   if (*string == '.')
      {
      string++;
      while ((*string != '\0') && ((char) toupper(*string) != 'E'))
         if (! isdigit((int) *string++)) return(FALSE);
      }

   if ((char) toupper(*string) == 'E')
      {
      string++;
      if ((*string == '+') || (*string == '-')) string++;
      while (*string != '\0')
         if (! isdigit((int) *string++)) return(FALSE);
      }

   return(TRUE);
}



int is_string_pos_number(string)
char *string;
{
/* First, determine if the beginning is a sign operator. */

   string = find_noblank(string);

   if (*string == '+') string++;
   else if (*string == '-') return FALSE;

   while ((*string != '\0') && (*string != '.') &&
          ((char) toupper(*string) != 'E'))
      if (! isdigit((int) *string++)) return(FALSE);

   if (*string == '.')
      {
      string++;
      while ((*string != '\0') && ((char) toupper(*string) != 'E'))
         if (! isdigit((int) *string++)) return(FALSE);
      }

   if ((char) toupper(*string) == 'E')
      {
      string++;
      if ((*string == '+') || (*string == '-')) string++;
      while (*string != '\0')
         if (! isdigit((int) *string++)) return(FALSE);
      }

   return(TRUE);
}


char *pad_string(string, count)
char *string;
int count;
{
int i;
char *ptr;
```



```
   ptr = string;
   i = strlen(string);

   for (string += strlen(string) ; i < count ; i++, *string++ = ' ');
   *string = '\0';

   return(ptr);
}



int blank_string(string, count)
char *string;
int count;
{
int i;

   for (i = 0 ; i < count ; i++) *string++ = ' ';
   return(count);
}
```

```
// ========================================================================
//                                  STUDY.C
// ========================================================================
// ========================================================================
//                         Opening and saving a study
// ========================================================================
// ========================================================================
//                            Revision History
// ========================================================================
//  09/30/94    Radian(RLB)             Initial Revision
// ========================================================================

#include <stdio.h>
#include <string.h>
#include <mem.h>
#include <windows.h>
#include "commdlg.h"

#include "landwin.h"
#include "landoper.h"

// Prototypes for defined in this function

int GetLandfillFilename(char *title, char *filename, int nSize);

/* Some local string space to maniupulate the filenames in */

static char szDirName[256], szFileTitle[256], szFilter[256];


BOOL GetOldStudy(void)
[
char caption[128];
HWND hWnd, ActiveTool;
HGLOBAL hOperData;
OPERDATASTRUCT *OperData;
int ret;
int LoadStudy(OPERDATASTRUCT *OperData);

   if (! GetLandfillFilename("Load a Landfill Study", StudyName, sizeof(StudyNam
e)))
      return FALSE;

   hWnd = CreateOperClientWindow("", CW_USEDEFAULT, CW_USEDEFAULT,
                              CW_USEDEFAULT, CW_USEDEFAULT,
                              WS_OVERLAPPEDWINDOW | WS_VISIBLE);


   OperData = (OPERDATASTRUCT *) GlobalLock(MDIGETOPERDATA(hWnd));
   if (OperData != NULL)
      |
      strcpy(OperData->Filename, StudyName);
      strcpy(caption, "Operating Parameters: ");
      strncpy(caption + strlen(caption), OperData->Filename,
              128 - strlen(caption) - 1);
      SetWindowText(hWnd, caption);

      ret = LoadStudy(OperData);

      GlobalUnlock(MDIGETOPERDATA(hWnd));
      ]

   if (! ret) DestroyWindow(hWnd);
//   else if (hWnd != NULL) ShowWindow(hWnd, SW_SHOWNORMAL);

   ActiveTool = SendMessage(RetrieveFrameWindow(), AM_GETACTIVETOOL, 0, 0);
```



```
   if (IsWindow(ActiveTool))
      SendMessage(GetWindow(ActiveTool, GW_CHILD), AM_UPDATEFIELDS, 0, hWnd);

   return TRUE;
}



BOOL GetNewStudy(void)
{
HWND hWnd, ActiveWindow, ActiveTool;
HGLOBAL hOperData;
OPERDATASTRUCT *OperData;
static int StudyIndex = 0;
void SetNewValues(HWND hWnd);

   sprintf(StudyName, "Operating Parameters: .\\LANDFILL.%03d", StudyIndex++);

   hWnd = CreateOperClientWindow(StudyName, CW_USEDEFAULT, CW_USEDEFAULT,
                              CW_USEDEFAULT, CW_USEDEFAULT,
                              WS_OVERLAPPEDWINDOW | WS_VISIBLE);

   OperData = (OPERDATASTRUCT *) GlobalLock(MDIGETOPERDATA(hWnd));

   if (OperData->Periods.hPeriod != NULL) GlobalFree(OperData->Periods.hPeriod);
   OperData->Periods.PeriodCount = 0;

   if (OperData->AirToxics.hAirToxic != NULL)
         GlobalFree(OperData->AirToxics.hAirToxic);
   OperData->AirToxics.AirToxicCount = 0;

   strcpy(OperData->Filename, StudyName);
   LoadModelDefaults(hWnd);

   OperData->Choices.LandfillType = Limits.LandfillType;
   OperData->Choices.NMOCChoice   = Limits.DefaultsMode;
   OperData->Choices.kChoice      = Limits.DefaultsMode;
   OperData->Choices.LOChoice     = Limits.DefaultsMode;
   OperData->Choices.DefaultsMode = Limits.DefaultsMode;

   SetModelDefaults(hWnd);

   GlobalUnlock(MDIGETOPERDATA(hWnd));

//   if (hWnd != NULL)  ShowWindow(hWnd, SW_SHOWNORMAL);

   ActiveTool = SendMessage(RetrieveFrameWindow(), AM_GETACTIVETOOL, 0, 0);
   if (IsWindow(ActiveTool))
      SendMessage(GetWindow(ActiveTool, GW_CHILD), AM_UPDATEFIELDS, 0, hWnd);

   return TRUE;
}



void SaveStudy(void)
{
HFILE file;
HGLOBAL hOperData;
HWND ActiveWindow;
OPERDATASTRUCT *OperData;
double *Period;
AIRTOXIC *AirToxic;
void FileUseError(void);
```

```
   ActiveWindow = MDIGETACTIVE();
   if (IsWindow(ActiveWindow)) hOperData = MDIGETOPERDATA(ActiveWindow);

   if (hOperData != NULL)
      OperData = (OPERDATASTRUCT *) GlobalLock(hOperData);

   if (OperData != NULL)
      {
      file = _lcreat((LPCSTR) OperData->Filename, 0);
      if (file == HFILE_ERROR)
         {
         FileUseError();
         return;
         }

      _lwrite(file, &(OperData->Model), sizeof(MODELPARAMETERS));
      _lwrite(file, &(OperData->Method2E), sizeof(METHOD2E));
      _lwrite(file, &(OperData->Parameters), sizeof(OPPARAMETERS));
      _lwrite(file, &(OperData->Choices), sizeof(PARAMETERINDEX));
      _lwrite(file, &(OperData->Periods.PeriodCount), sizeof(int));

      if (OperData->Periods.PeriodCount)
         {
         Period = (double *) GlobalLock(OperData->Periods.hPeriod);
         if (Period != NULL)
            {
            _lwrite(file, Period, sizeof(double) * OperData->Periods.PeriodCount
);
            GlobalUnlock(OperData->Periods.hPeriod);
            }
         }

      _lwrite(file, &(OperData->AirToxics.AirToxicCount), sizeof(int));

      if (OperData->AirToxics.AirToxicCount)
         {
         AirToxic = (AIRTOXIC *) GlobalLock(OperData->AirToxics.hAirToxic);
         if (AirToxic != NULL)
            {
            _lwrite(file, AirToxic, sizeof(AIRTOXIC)
                        * OperData->AirToxics.AirToxicCount);
            GlobalUnlock(OperData->AirToxics.hAirToxic);
            }
         }

      _lclose(file);

      GlobalUnlock(hOperData);
      }
}



int LoadStudy(OPERDATASTRUCT *OperData)
{
HFILE file;
double *Period;
AIRTOXIC *AirToxic;
void FileUseError(void);

   if (OperData == NULL) return 0;

   file = _lopen((LPCSTR) OperData->Filename, READ);
   if (file == HFILE_ERROR)
      {
```



```
      FileUseError();
      return 0;
      }

   _lread(file, &(OperData->Model), sizeof(MODELPARAMETERS));
   _lread(file, &(OperData->Method2E), sizeof(METHOD2E));
   _lread(file, &(OperData->Parameters), sizeof(OPPARAMETERS));
   _lread(file, &(OperData->Choices), sizeof(PARAMETERINDEX));
   _lread(file, &(OperData->Periods.PeriodCount), sizeof(int));

   if (OperData->Periods.PeriodCount)
      {
      OperData->Periods.hPeriod = GlobalAlloc(GHND,
                        OperData->Periods.PeriodCount * sizeof(double));
      if (OperData->Periods.hPeriod == NULL)
         {
         MemAllocError();
         _lclose(file);

         return 0;
         }

      Period = (double *) GlobalLock(OperData->Periods.hPeriod);
      if (Period != NULL)
         {
         _lread(file, Period, sizeof(double) * OperData->Periods.PeriodCount);
         GlobalUnlock(OperData->Periods.hPeriod);
         }
      }

   _lread(file, &(OperData->AirToxics.AirToxicCount), sizeof(int));

   if (OperData->AirToxics.AirToxicCount)
      {
      OperData->AirToxics.hAirToxic = GlobalAlloc(GHND,
                        OperData->AirToxics.AirToxicCount * sizeof(AIRTOXIC));
      if (OperData->AirToxics.hAirToxic == NULL)
         {
         MemAllocError();
         _lclose(file);

         return 0;
         }

      AirToxic = (AIRTOXIC *) GlobalLock(OperData->AirToxics.hAirToxic);
      if (AirToxic != NULL)
         {
         _lread(file, AirToxic, sizeof(AIRTOXIC)
                     * OperData->AirToxics.AirToxicCount);
         GlobalUnlock(OperData->AirToxics.hAirToxic);
         }
      }

   _lclose(file);

   return 1;
}



void SaveAsStudy(void)
{
HGLOBAL hOperData, hOperTest;
HWND ActiveWindow, WindowPtr;
```

```
OPERDATASTRUCT *OperData;
char *ptr;
char caption[128];

   if (! GetLandfillFilename("Save As Landfill Study",
                          StudyName, sizeof(StudyName))) return;

   ActiveWindow = MDIGETACTIVE();

   if (IsWindow(ActiveWindow)) hOperData = MDIGETOPERDATA(ActiveWindow);
   else return;

   if (hOperData == NULL) return;

   OperData = (OPERDATASTRUCT *) GlobalLock(hOperData);

   strcpy(OperData->Filename, StudyName);

   SaveStudy();

// This checks all MDI client windows to see if they are members of this
// study.  If so change their titles

   for (WindowPtr = GetWindow(ActiveWindow, GW_HWNDFIRST);
        WindowPtr != NULL ;
        WindowPtr = GetWindow(WindowPtr, GW_HWNDNEXT))
      {
      hOperTest = MDIGETOPERDATA(WindowPtr);

// Retitle the windows with the output filename

      if (hOperTest == hOperData)
         {
         switch (WhichMDIType(WindowPtr))
            {
            case OPERWINDOW:
               strcpy(caption, "Operating Parameters: ");
               strncpy(caption + strlen(caption), OperData->Filename,
                       128 - strlen(caption) - 1);
               SetWindowText(WindowPtr, caption);
               break;

            case GRAPHWINDOW:
               strcpy(caption, "Graphics: ");
               strncpy(caption + strlen(caption), OperData->Filename,
                       128 - strlen(caption) - 1);
               SetWindowText(WindowPtr, caption);
               break;


            default:
               break;
            }
         }
      }

   GlobalUnlock(hOperData);
}



void FileUseError(void)
{
```



```
   MessageBox(NULL, "Unable to Open File", "File Use Error",
                MB_ICONSTOP | MB_OK);
}



int GetLandfillFilename(char *title, char *filename, int nSize)
{
OPENFILENAME ofn;
UINT  i, cbString;
char  chReplace;    /* string separator for szFilter */
int ret;

        *filename = '\0';
   GetSystemDirectory(szDirName, sizeof(szDirName));

        if ((cbString = LoadString(RetrieveAppInstance(), IDS_FILTERSTRING,
                szFilter, sizeof(szFilter))) == 0) return 0;

        chReplace = szFilter[cbString - 1]; /* retrieve wildcard */

   for (i = 0; szFilter[i] != '\0'; i++)
                if (szFilter[i] == chReplace) szFilter[i] = '\0';

/* Set all structure members to zero. */

   memset(&ofn, 0, sizeof(OPENFILENAME));

   ofn.lStructSize = sizeof(OPENFILENAME);
   ofn.hwndOwner = RetrieveFrameWindow();
   ofn.lpstrFilter = szFilter;
   ofn.nFilterIndex = 1;
        ofn.lpstrFile= filename;
   ofn.nMaxFile = nSize;
   ofn.lpstrFileTitle = szFileTitle;
   ofn.nMaxFileTitle = sizeof(szFileTitle);
   ofn.lpstrInitialDir = NULL;
   ofn.lpstrTitle = (LPSTR) title;
   ofn.Flags = OFN_PATHMUSTEXIST;
   ofn.lpstrDefExt = "PRM";

   ret = GetOpenFileName(&ofn);

        return ret;
}
```

```
#include <windows.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <ctype.h>

#include "landwin.h"
#include "str_ops.h"

#define UP_3D    1
#define DOWN_3D  2

void Draw3DFrame(HWND hWnd, int iStyle);



LRESULT CALLBACK FPNumberEditProc(HWND hWnd, UINT iMessage, WPARAM wParam,
                                  LPARAM lParam)
{
char string[30];
LRESULT ret;
void UnCustomizeControl(HWND hWnd);
LRESULT CALLBACK CallSCProc(HWND hWnd, UINT iMessage, WORD wParam,
                            DWORD lParam);
NCCALCSIZE_PARAMS *lpncsp = (NCCALCSIZE_PARAMS *) lParam;
WINDOWPOS *WinPos;

   switch (iMessage)
      {
      case WM_NCDESTROY:
         UnCustomizeControl(hWnd);
         return TRUE;

      case WM_CHAR:
// Only pass the windows default procedure the characters we are interested
// in keeping

         memset(string, 0, sizeof(string));
         SendMessage(hWnd, WM_GETTEXT, sizeof(string), (long) string);

         *(string + strlen(string)) = (char) wParam;

         if (isprint((char) wParam))
            {
            if (is_string_pos_number(string))
                 return CallSCProc(hWnd, iMessage, wParam, lParam);
            else return TRUE;
            }

         return CallSCProc(hWnd, iMessage, wParam, lParam);

      case WM_NCPAINT:
         Draw3DFrame(hWnd, DOWN_3D);
         return TRUE;

      case WM_NCCALCSIZE:
         if (! (BOOL) wParam) return FALSE;

         WinPos = lpncsp->lppos;

         CopyRect((RECT *) lpncsp->rgrc) + 2, lpncsp->rgrc);

         lpncsp->rgrc[1].left += 2;
         lpncsp->rgrc[1].right -= 2;
         lpncsp->rgrc[1].top += 2;
         lpncsp->rgrc[1].bottom -= 2;
```



```
         lpncsp->rgrc[2].left += 2;
         lpncsp->rgrc[2].right -= 2;
         lpncsp->rgrc[2].top += 2;
         lpncsp->rgrc[2].bottom -= 2;

         CopyRect(lpncsp->rgrc, ((RECT *) lpncsp->rgrc) + 2);

         lpncsp->rgrc[0].left = WinPos->x + 2;
         lpncsp->rgrc[0].top = WinPos->y + 2;
         lpncsp->rgrc[0].right = WinPos->cx - 2;
         lpncsp->rgrc[0].bottom = WinPos->cy - 2;

         return WVR_VALIDRECTS & WVR_HREDRAW & WVR_VREDRAW;

      case WM_PAINT:
         CallSCProc(hWnd, iMessage, wParam, lParam);
         Draw3DFrame(hWnd, DOWN_3D);
         return FALSE;

      default:
         return CallSCProc(hWnd, iMessage, wParam, lParam);
      }
}



LRESULT CALLBACK IntNumberEditProc(HWND hWnd, UINT iMessage, WPARAM wParam,
                                   LPARAM lParam)
{
LRESULT ret;
LRESULT CALLBACK CallSCProc(HWND hWnd, UINT iMessage, WORD wParam,
                            DWORD lParam);

   switch (iMessage)
      {
      case WM_NCDESTROY:
         UnCustomizeControl(hWnd);
         return TRUE;

      case WM_CHAR:
         if (isalnum((char) wParam))
            {
            if (isdigit((char) wParam))
               return CallSCProc(hWnd, iMessage, wParam, lParam);
            else return TRUE;
            }
         else return CallSCProc(hWnd, iMessage, wParam, lParam);

      case WM_PAINT:
         ret = CallSCProc(hWnd, iMessage, wParam, lParam);
         Draw3DFrame(hWnd, DOWN_3D);
         return ret;

      default:
         return CallSCProc(hWnd, iMessage, wParam, lParam);
      }
}



void CustomizeControl(HWND hWndDlg, int ControlID, FARPROC Procedure)
{
```

```
DWORD ProcLong;

   CustomizeWindowControl(GetDlgItem(hWndDlg, ControlID), Procedure);
]



void CustomizeWindowControl(HWND hWnd, FARPROC Procedure)
{
DWORD ProcLong;

// To customize the control, we get the default windows procedure from
// the window structure and then save it as a property using a couple
// of new property types.  Then, we insure the new one.

   ProcLong = GetWindowLong(hWnd, GWL_WNDPROC);
   SetProp(hWnd, "SCHookHI", HIWORD(ProcLong));
   SetProp(hWnd, "SCHookLO", LOWORD(ProcLong));

   SetWindowLong(hWnd, GWL_WNDPROC, (LONG) Procedure);
]



void UnCustomizeControl(HWND hWndDlg)
{
WORD ProcLO, ProcHI;

   ProcLO = GetProp(hWndDlg, "SCHookLO");
   ProcHI = GetProp(hWndDlg, "SCHookHI");

   SetWindowLong(hWndDlg, GWL_WNDPROC, MAKELONG(ProcLO, ProcHI));

   RemoveProp(hWndDlg, "SCHookLO");
   RemoveProp(hWndDlg, "SCHookHI");
]



LRESULT CALLBACK CallSCProc(HWND hWnd, UINT iMessage, WORD wParam,
                            DWORD lParam)
{
FARPROC WndProc;
WORD ProcLO, ProcHI;

   ProcLO = GetProp(hWnd, "SCHookLO");
   ProcHI = GetProp(hWnd, "SCHookHI");

   WndProc = (FARPROC) MAKELONG(ProcLO, ProcHI);

   return CallWindowProc(WndProc, hWnd, iMessage, wParam, lParam);
}


void Draw3DFrame(HWND hWnd, int iStyle)
{
HDC hDC;
HPEN hOldPen, hLTOuterPen, hLTInnerPen, hRBOuterPen, hRBInnerPen;
// HPEN hInnerPen;
RECT r;

   switch (iStyle)
      {
```



```
      case UP_3D:
         hRBOuterPen = CreatePen(PS_SOLID, 1, RGB(64, 64, 64));
         hRBInnerPen = CreatePen(PS_SOLID, 1, RGB(0, 0, 0));
         hLTInnerPen = CreatePen(PS_SOLID, 1, RGB(128, 128, 128));
         hLTOuterPen = CreatePen(PS_SOLID, 1, RGB(255, 255, 255));
         break;

      case DOWN_3D:
         hLTOuterPen = CreatePen(PS_SOLID, 1, RGB(64, 64, 64));
         hLTInnerPen = CreatePen(PS_SOLID, 1, RGB(0, 0, 0));
         hRBInnerPen = CreatePen(PS_SOLID, 1, RGB(128, 128, 128));
         hRBOuterPen = CreatePen(PS_SOLID, 1, RGB(255, 255, 255));
         break;
      }

   hDC = GetWindowDC(hWnd);
   GetWindowRect(hWnd, &r);

   hOldPen = SelectObject(hDC, hLTInnerPen);

   MoveTo(hDC, 1, (r.bottom - r.top - 3));
   LineTo(hDC, 1, 1);
   LineTo(hDC, (r.right - r.left - 2), 1);

   SelectObject(hDC, hRBInnerPen);

   MoveTo(hDC, (r.right - r.left - 2), 1);
   LineTo(hDC, (r.right - r.left - 2), (r.bottom - r.top - 2));
   LineTo(hDC, 0, (r.bottom - r.top - 2));

   SelectObject(hDC, hLTOuterPen);

   MoveTo(hDC, 0, (r.bottom - r.top - 3));
   LineTo(hDC, 0, 0);
   LineTo(hDC, (r.right - r.left - 2), 0);

   SelectObject(hDC, hRBOuterPen);
   MoveTo(hDC, (r.right - r.left - 1), 0);
   LineTo(hDC, (r.right - r.left - 1), (r.bottom - r.top - 1));
   LineTo(hDC, -1, (r.bottom - r.top - 1));

   SelectObject(hDC, hOldPen);

   ReleaseDC(hWnd, hDC);
   DeleteObject(hLTOuterPen);
   DeleteObject(hLTInnerPen);
   DeleteObject(hRBOuterPen);
   DeleteObject(hRBInnerPen);
}
```

```
// ==========================================================================
//                                TEXTFILE.C
// ==========================================================================
// ==========================================================================
//                     Handles report output to a file
// ==========================================================================
// ==========================================================================
//                            Revision History
// ==========================================================================
//  09/30/94    Radian(RLB)              Initial Revision
// ==========================================================================

#include <stdio.h>
#include <mem.h>
#include <windows.h>
#include "commdlg.h"

#include "landwin.h"
#include "textrpt.h"


extern char StudyName[MAXPATH];

void DoTextOutput(HWND hWnd, REPORTSTRUCT *ReportStruct)
{
HFILE hFile;
int i;
char (*string)[PAGE_WIDTH + 5];
char OutputPath[MAXPATH];
int GetTextOutputFilename(HINSTANCE hInst, HWND hWnd, char *filename, int nSize)
;

   if (ReportStruct == NULL) return;

   string = (char (*)[PAGE_WIDTH + 5]) GlobalLock(ReportStruct->hReportText);

   if (GetTextOutputFilename(RetrieveAppInstance(), hWnd, OutputPath,
       MAXPATH))
      {
      hFile = _lcreat((LPCSTR) OutputPath, 0);

      for (i = 0 ; i < ReportStruct->LineCount ; i++)
         {
         _lwrite(hFile, string[i], PAGE_WIDTH);
         _lwrite(hFile, "\r\n", 2);
         }

      _lclose(hFile);
      }

   GlobalUnlock(ReportStruct->hReportText);
}


int GetTextOutputFilename(HINSTANCE hInst, HWND hWnd, char *filename, int nSize)
{
OPENFILENAME ofn;
char szDirName[256];
char szFileTitle[256];
UINT  i, cbString;
char  chReplace;    /* string separator for szFilter */
char  szFilter[256];
int ret;

        *filename = '\0';
   GetSystemDirectory(szDirName, sizeof(szDirName));
```



```
        if ((cbString = LoadString(hInst, IDS_TEXTFILTER,
                szFilter, sizeof(szFilter))) == 0) return 0;

        chReplace = szFilter[cbString - 1]; /* retrieve wildcard */

   for (i = 0; szFilter[i] != '\0'; i++)
                if (szFilter[i] == chReplace) szFilter[i] = '\0';

/* Set all structure members to zero. */

   memset(&ofn, 0, sizeof(OPENFILENAME));

   ofn.lStructSize = sizeof(OPENFILENAME);
   ofn.hwndOwner = hWnd;
   ofn.lpstrFilter = szFilter;
   ofn.nFilterIndex = 1;
        ofn.lpstrFile= filename;
   ofn.nMaxFile = nSize;
   ofn.lpstrFileTitle = szFileTitle;
   ofn.nMaxFileTitle = sizeof(szFileTitle);
   ofn.lpstrInitialDir = NULL;
   ofn.lpstrTitle = "Save Text to File";
   ofn.Flags = OFN_PATHMUSTEXIST;
   ofn.lpstrDefExt = "TXT";

   ret = GetOpenFileName(&ofn);

        return ret;
}
```

```
#include <stdio.h>
#include <mem.h>
#include <math.h>
#include <windows.h>
#include <string.h>

#include "commdlg.h"

#include "landwin.h"
#include "results.h"
#include "textrpt.h"

HWND CreateTextClientWindow(char *title, int x, int y, int width, int height,
                            DWORD style)
{
MDICREATESTRUCT MDIcs;
HWND hWnd;

/* Set up the MDI Structure to create the client window.  lParam contains */
/* an id that is stored in the window that indicates the type of MDI */
/* window this is and which WndProc needs to be invoked. */

   MDIcs.szClass = "TEXTWINDOW";
   MDIcs.szTitle = title;
   MDIcs.hOwner = RetrieveAppInstance();
   MDIcs.x = x;
   MDIcs.y = y;
   MDIcs.cx = width;
   MDIcs.cy = height;
   MDIcs.style = style;
   MDIcs.lParam = 0;

   hWnd = SendMessage(RetrieveClientWindow(), WM_MDICREATE, 0, (LPARAM) &MDIcs);

   return hWnd;
}



long FAR PASCAL TextWndProc(HWND hWnd, WORD iMessage, WPARAM wParam, LPARAM lPar
am)
{
HDC hdc;
HFONT hFont, hOldFont;
PAINTSTRUCT lpps;
TEXTMETRIC tm;
RECT rect;
HGLOBAL hPtr;
OPERDATASTRUCT *OperData;
REPORTSTRUCT *ReportStruct;
RESULTS *Results;
HGLOBAL hTextRpt, hOperData, hResults, hResultsNew;
HLOCAL hlf;
LOGFONT *plf;
int WindowLines, WindowChars;
int OffsetY, OffsetX, i;
int OldPos, NewPos;
int MinPos, MaxPos;
int ret;
char (*string)[PAGE_WIDTH + 5], *clip;
void DoTextOutput(HWND hWnd, REPORTSTRUCT *ReportStruct);
void DoTextPrintOutput(HINSTANCE hInst, HWND hWnd);

     switch (iMessage)
        {
        case WM_CREATE:
```



```
           SetWindowLong(hWnd, 4, (long) NULL);  // Holds the ReportStruct infor
mation
           SetWindowLong(hWnd, 8, (long) NULL);  // Holds the Results informatio
n

           hdc = GetDC(hWnd);

           hlf = LocalAlloc(LHND, sizeof(LOGFONT));
           if (hlf == NULL) return FALSE;

           plf = (LOGFONT *) LocalLock(hlf);
           if (plf == NULL) return FALSE;

           plf->lfHeight = -12;
           lstrcpy(plf->lfFaceName, "Courier New");

           hFont = CreateFontIndirect(plf);
           hOldFont = SelectObject(hdc, hFont);
           SetTextAlign(hdc, TA_LEFT);

           DeleteObject(hOldFont);

           LocalUnlock(hlf);
           LocalFree(hlf);

           ReleaseDC(hWnd, hdc);

           return TRUE;

        case WM_MDIACTIVATE:
           SendMessage(RetrieveFrameWindow(), AM_WINDOWACTIVE, 0, 0);
           return TRUE;

        case WM_SIZE:
/* Determine how many lines are available to be displayed */
           hdc = GetDC(hWnd);
           GetTextMetrics(hdc, &tm);
           ReleaseDC(hWnd, hdc);

           WindowLines = HIWORD(lParam) / (tm.tmHeight + tm.tmExternalLeading);
           WindowChars = LOWORD(lParam) / tm.tmAveCharWidth;

           hTextRpt = GetWindowLong(hWnd, 4);
           if (hTextRpt != NULL)
              {
              ReportStruct = (REPORTSTRUCT *) GlobalLock(hTextRpt);

              SetScrollRange(hWnd, SB_VERT, 0,
                             ReportStruct->LineCount - WindowLines - 1,
                               FALSE);

              if (WindowChars < PAGE_WIDTH)
                 SetScrollRange(hWnd, SB_HORZ, 0, PAGE_WIDTH - WindowChars, FALS
E);
              else SetScrollRange(hWnd, SB_HORZ, 0, 0, TRUE);


              GlobalUnlock(hTextRpt);
              }

           return FALSE;

        case AM_GETOPERHANDLE:
           return (long) GetWindowLong(hWnd, 0);

        case AM_VALUESCHANGED:
```

```
// Oper values changed.  Have to reinitiate a calculation

            hOperData = GetWindowLong(hWnd, 0);
            hTextRpt  = GetWindowLong(hWnd, 4);
            hResults  = GetWindowLong(hWnd, 8);

            if (hResults != NULL)
               {
               Results = (RESULTS *) GlobalLock(hResults);

               if (ValidateData())
                  {
                  hResultsNew = CalculationModule(Results->AirToxicType);
                  if (hResultsNew == NULL)
                     {
                     GlobalUnlock(hResults);
                     return TRUE;
                     }
                  }
               else
                  {
                  GlobalUnlock(hResults);
                  return TRUE;
                  }

               if (Results->hRate != NULL) GlobalFree(Results->hRate);

               GlobalUnlock(hResults);
               GlobalFree(hResults);

               SetWindowLong(hWnd, 8, hResultsNew);

               hResults  = GetWindowLong(hWnd, 8);
               Results = (RESULTS *) GlobalLock(hResults);

               if (hOperData != NULL)
                  OperData = (OPERDATASTRUCT *) GlobalLock(hOperData);

               if ((hOperData != NULL) && (hResults != NULL))
                  {
                  if (hTextRpt != NULL)
                     {
                     ReportStruct = (REPORTSTRUCT *) GlobalLock(hTextRpt);
                     if (ReportStruct->hReportText != NULL)
                        GlobalFree(ReportStruct->hReportText);
                     GlobalUnlock(hTextRpt);
                     GlobalFree(hTextRpt);
                     }

                  hTextRpt = BuildReport(OperData, Results);

                  ReportStruct = (REPORTSTRUCT *) GlobalLock(hTextRpt);

                  hdc = GetDC(hWnd);
                  GetTextMetrics(hdc, &tm);
                  ReleaseDC(hWnd, hdc);

                  GetClientRect(hWnd, &rect);

                  WindowLines = rect.bottom / (tm.tmHeight + tm.tmExternalLeading
);
                  WindowChars = rect.right / tm.tmAveCharWidth;

                  SetScrollRange(hWnd, SB_VERT, 0,
                                 ReportStruct->LineCount - WindowLines - 1,
                                 FALSE);
```



```
                  SetScrollPos(hWnd, SB_VERT, 0, TRUE);

                  if (WindowChars < PAGE_WIDTH)
                     SetScrollRange(hWnd, SB_HORZ, 0, PAGE_WIDTH - WindowChars, F
ALSE);
                  else SetScrollRange(hWnd, SB_HORZ, 0, 0, TRUE);


                  SetScrollPos(hWnd, SB_HORZ, 0, TRUE);

                  SetWindowLong(hWnd, 4, (long) hTextRpt);
                  }

               if (hOperData != NULL) GlobalUnlock(hOperData);
               if (hResults != NULL)  GlobalUnlock(hResults);
               }

            InvalidateRect(hWnd, NULL, TRUE);
            return TRUE;

         case AM_COPY:
         case AM_CUT:
            if (OpenClipboard(hWnd))
               {
               EmptyClipboard();

               hTextRpt  = GetWindowLong(hWnd, 4);

               if (hTextRpt != NULL)
                  {
                  ReportStruct = (REPORTSTRUCT *) GlobalLock(hTextRpt);

                  hPtr = GlobalAlloc(GHND, ReportStruct->LineCount * (PAGE_WIDTH
+ 5) + 1);
                  if (hPtr == NULL)
                     {
                     GlobalUnlock(hTextRpt);
                     return TRUE;
                     }

                  string = (char (*)[PAGE_WIDTH + 5]) GlobalLock(ReportStruct->hR
eportText);
                  clip = (char *) GlobalLock(hPtr);

                  for (i = 0 ; i < ReportStruct->LineCount ; i++)
                     {
                     strcpy(clip, string[i]);
                     strcat(clip, "\r\n");

                     clip += strlen(string[i]) + 2;
                     }

                  GlobalUnlock(ReportStruct->hReportText);
                  GlobalUnlock(hTextRpt);
                  GlobalUnlock(hPtr);

                  if (hPtr != NULL)
                     {
                     SetClipboardData(CF_TEXT, hPtr);
                     CloseClipboard();
                     }
                  }
               }

            return TRUE;
```

```
        case AM_SAVE:
           if ((hTextRpt = (HGLOBAL) GetWindowLong(hWnd, 4)) != NULL)
              {
              ReportStruct = (REPORTSTRUCT *) GlobalLock(hTextRpt);
              DoTextOutput(hWnd, ReportStruct);
              GlobalUnlock(hTextRpt);
              }

           return TRUE;

        case AM_PRINT:
           DoTextPrintOutput(RetrieveAppInstance(), hWnd);
           return TRUE;

        case WM_VSCROLL:
           hdc = GetDC(hWnd);
           GetTextMetrics(hdc, &tm);
           ReleaseDC(hWnd, hdc);

           switch (wParam)
              {
              case SB_BOTTOM:
                 GetScrollRange(hWnd, SB_VERT, &MinPos, &MaxPos);
                 if (GetScrollPos(hWnd, SB_VERT) != MaxPos)
                    {
                    SetScrollPos(hWnd, SB_VERT, MaxPos, TRUE);
                    InvalidateRect(hWnd, NULL, TRUE);
                    }
                 break;

              case SB_LINEDOWN:
                 GetScrollRange(hWnd, SB_VERT, &MinPos, &MaxPos);
                 if ((ret = GetScrollPos(hWnd, SB_VERT)) < MaxPos)
                    {
                    GetClientRect(hWnd, &rect);
                    WindowLines = rect.bottom / (tm.tmHeight + tm.tmExternalLead
ing);
                    rect.bottom = WindowLines * (tm.tmHeight + tm.tmExternalLead
ing);

                    ScrollWindow(hWnd, 0, -(tm.tmHeight + tm.tmExternalLeading),
                                 NULL, &rect);
                    SetScrollPos(hWnd, SB_VERT, ret + 1, TRUE);
                    UpdateWindow(hWnd);
                    }

                 break;

              case SB_LINEUP:
                 if ((ret = GetScrollPos(hWnd, SB_VERT)) > 0)
                    {
                    GetClientRect(hWnd, &rect);
                    WindowLines = rect.bottom / (tm.tmHeight + tm.tmExternalLead
ing);
                    rect.bottom = WindowLines * (tm.tmHeight + tm.tmExternalLead
ing);

                    ScrollWindow(hWnd, 0, (tm.tmHeight + tm.tmExternalLeading),
                                 NULL, &rect);
                    SetScrollPos(hWnd, SB_VERT, ret - 1, TRUE);
                    UpdateWindow(hWnd);
                    }

                 break;
```



```
              case SB_PAGEDOWN:
                 GetScrollRange(hWnd, SB_VERT, &MinPos, &MaxPos);
                 if ((ret = GetScrollPos(hWnd, SB_VERT)) < MaxPos)
                    {
                    GetClientRect(hWnd, &rect);
                    WindowLines = rect.bottom / (tm.tmHeight + tm.tmExternalLead
ing);
                    rect.bottom = WindowLines * (tm.tmHeight + tm.tmExternalLead
ing);

                    if ((ret + WindowLines) > MaxPos)
                       SetScrollPos(hWnd, SB_VERT, MaxPos, TRUE);
                    else
                       SetScrollPos(hWnd, SB_VERT, ret + WindowLines, TRUE);

                    InvalidateRect(hWnd, NULL, TRUE);
                    }
                 break;

              case SB_PAGEUP:
                 if ((ret = GetScrollPos(hWnd, SB_VERT)) > 0)
                    {
                    GetClientRect(hWnd, &rect);
                    WindowLines = rect.bottom / (tm.tmHeight + tm.tmExternalLead
ing);
                    rect.bottom = WindowLines * (tm.tmHeight + tm.tmExternalLead
ing);

                    if ((ret - WindowLines) < 0)
                       SetScrollPos(hWnd, SB_VERT, 0, TRUE);
                    else
                       SetScrollPos(hWnd, SB_VERT, ret - WindowLines, TRUE);

                    InvalidateRect(hWnd, NULL, TRUE);
                    }
                 break;

              case SB_THUMBTRACK:
                 GetClientRect(hWnd, &rect);
                 WindowLines = rect.bottom / (tm.tmHeight + tm.tmExternalLeading
);
                 rect.bottom = WindowLines * (tm.tmHeight + tm.tmExternalLeading
);

                 ret = GetScrollPos(hWnd, SB_VERT);
                 ScrollWindow(hWnd, 0, (ret - LOWORD(lParam))
                              * (tm.tmHeight + tm.tmExternalLeading),
                              NULL, &rect);
                 SetScrollPos(hWnd, SB_VERT, LOWORD(lParam), FALSE);
                 UpdateWindow(hWnd);
                 break;

              case SB_THUMBPOSITION:
                 ret = GetScrollPos(hWnd, SB_VERT);
                 ScrollWindow(hWnd, 0, (ret - LOWORD(lParam))
                              * (tm.tmHeight + tm.tmExternalLeading),
                              NULL, NULL);
                 SetScrollPos(hWnd, SB_VERT, LOWORD(lParam), TRUE);
                 UpdateWindow(hWnd);
                 break;
              }

           return TRUE;

        case WM_HSCROLL:
           hdc = GetDC(hWnd);
```

```
        GetTextMetrics(hdc, &tm);
        ReleaseDC(hWnd, hdc);

        switch (wParam)
            {
            case SB_LINEDOWN:
                GetScrollRange(hWnd, SB_HORZ, &MinPos, &MaxPos);
                if ((ret = GetScrollPos(hWnd, SB_HORZ)) < MaxPos)
                    {
                    ScrollWindow(hWnd, -tm.tmAveCharWidth, 0, NULL, NULL);
                    SetScrollPos(hWnd, SB_HORZ, ret + 1, TRUE);
                    UpdateWindow(hWnd);
                    }
                break;

            case SB_LINEUP:
                if ((ret = GetScrollPos(hWnd, SB_HORZ)) > 0)
                    {
                    ScrollWindow(hWnd, tm.tmAveCharWidth, 0, NULL, NULL);
                    SetScrollPos(hWnd, SB_HORZ, ret - 1, TRUE);
                    UpdateWindow(hWnd);
                    }
                break;

            case SB_PAGEDOWN:
                GetScrollRange(hWnd, SB_VERT, &MinPos, &MaxPos);
                if ((ret = GetScrollPos(hWnd, SB_HORZ)) < MaxPos)
                    {
                    GetClientRect(hWnd, &rect);
                    WindowChars = rect.right / tm.tmAveCharWidth;

                    if ((ret + WindowChars) > MaxPos)
                        SetScrollPos(hWnd, SB_HORZ, MaxPos, TRUE);
                    else
                        SetScrollPos(hWnd, SB_HORZ, ret + WindowChars, TRUE);

                    InvalidateRect(hWnd, NULL, TRUE);
                    }
                break;

            case SB_PAGEUP:
                if ((ret = GetScrollPos(hWnd, SB_HORZ)) > 0)
                    {
                    GetClientRect(hWnd, &rect);
                    WindowChars = rect.right / tm.tmAveCharWidth;

                    if ((ret - WindowChars) > 0)
                        SetScrollPos(hWnd, SB_HORZ, 0, TRUE);
                    else
                        SetScrollPos(hWnd, SB_HORZ, ret - WindowChars, TRUE);

                    InvalidateRect(hWnd, NULL, TRUE);
                    }
                break;

            case SB_THUMBTRACK:
                ret = GetScrollPos(hWnd, SB_HORZ);
                ScrollWindow(hWnd, (ret - LOWORD(lParam)) * tm.tmAveCharWidth,
                             0, NULL, NULL);
                SetScrollPos(hWnd, SB_HORZ, LOWORD(lParam), FALSE);
                UpdateWindow(hWnd);
                break;

            case SB_THUMBPOSITION:
                ret = GetScrollPos(hWnd, SB_HORZ);
                ScrollWindow(hWnd, (ret - LOWORD(lParam)) * tm.tmAveCharWidth,
```



```
                             0, NULL, NULL);
                SetScrollPos(hWnd, SB_HORZ, LOWORD(lParam), TRUE);
                UpdateWindow(hWnd);
                break;
                }

            return TRUE;

        case WM_PAINT:

            BeginPaint(hWnd, &lpps);
            hdc = lpps.hdc;

            if ((hTextRpt = (HGLOBAL) GetWindowLong(hWnd, 4)) != NULL)
                {
                ReportStruct = (REPORTSTRUCT *) GlobalLock(hTextRpt);
                string = (char (*)[PAGE_WIDTH + 5]) GlobalLock(ReportStruct->hRepo
rtText);

/* Determine how many lines are available to be displayed */

                GetClientRect(hWnd, &rect);
                GetTextMetrics(hdc, &tm);

                WindowLines = rect.bottom / (tm.tmHeight + tm.tmExternalLeading);
                OffsetY = GetScrollPos(hWnd, SB_VERT);
                OffsetX = GetScrollPos(hWnd, SB_HORZ);

                for (i = 0 ;
                     (i < WindowLines) && ((i + OffsetY) < ReportStruct->LineCount
) ;
                     i++)
                    {
                    TextOut(hdc, 0, i * (tm.tmHeight + tm.tmExternalLeading),
                            string[i + OffsetY] + OffsetX,
                            strlen(string[i + OffsetY] + OffsetX));
                    }

                GlobalUnlock(ReportStruct->hReportText);
                GlobalUnlock(hTextRpt);
                }

            EndPaint(hWnd, &lpps);
            return TRUE;

        case WM_DESTROY:
            hPtr = (HGLOBAL) GetWindowLong(hWnd, 4);
            if (hPtr != NULL) GlobalFree(hPtr);

            hPtr = (HGLOBAL) GetWindowLong(hWnd, 0);
            if (hPtr != NULL) GlobalFree(hPtr);

            PostMessage(RetrieveFrameWindow(), AM_WINDOWACTIVE, 0, 0L);
            break;
        }

    return(DefMDIChildProc(hWnd, iMessage, wParam, lParam));
}
```

```
// ============================================================================
//                                TEXTPRNT.C
// ============================================================================
// ============================================================================
//                Handles text report output to the printer
// ============================================================================
// ============================================================================
//                            Revision History
// ============================================================================
//  09/30/94     Radian(RLB)              Initial Revision
// ============================================================================

#include <stdio.h>
#include <mem.h>
#include <string.h>
#include <windows.h>
#include <print.h>
#include "commdlg.h"

#include "landwin.h"
#include "textrpt.h"


typedef struct {
                                        char lpPrintDevice[40];
                                        char lpPrintDriver[20];
                                        char lpPrintPort[20];
                                        } PRINTERINFOSTRUCT;

static DEVMODE DMOut, DMIn;
static PRINTERINFOSTRUCT pinfo;
static char DeviceDriver[80];

void GetIndexedPrinter(PRINTERINFOSTRUCT *pinfo, int index);


void DoTextPrintOutput(HINSTANCE hInst, HWND hWnd)
{
HGLOBAL hPtr;
REPORTSTRUCT *ReportStruct;
char (*string)[PAGE_WIDTH + 5];
char HeaderString[85];
HCURSOR hOldCursor;
RECT rect;
HDC hdc;
DLGPROC dlgproc;
int index, lines, MaxLines;
int Page = 1;
DOCINFO DocInfo;
HLOCAL hlf;
LOGFONT *plf;
HFONT hFont, hOldFont;
TEXTMETRIC tm;
void GetIndexedPrinter(PRINTERINFOSTRUCT *pinfo, int index);
BOOL CALLBACK PrintDlgProc(HWND hWndDlg, UINT msg, WPARAM wParam,

 LPARAM lParam);
void AtSay(char *String, int loc, char *Text);
void InitializeOrientation(int Orient);

        dlgproc = (DLGPROC) MakeProcInstance(PrintDlgProc, hInst);
        index = DialogBox(hInst, "DIALOG_PRINT", hWnd, dlgproc);
        FreeProcInstance((FARPROC) dlgproc);

   if (index == -1) return;
```



```
        GetIndexedPrinter(&pinfo, index);

        hdc = CreateDC(pinfo.lpPrintDriver,
                                                pinfo.lpPrintDevice,
                                                pinfo.lpPrintPort, (const void f
ar *) &DMOut);

        if (hdc != NULL)
                {
        DocInfo.cbSize = sizeof(DOCINFO);
           DocInfo.lpszDocName = "Text Print";
                DocInfo.lpszOutput = NULL;

        StartDoc(hdc, &DocInfo);

        rect.top = 0;
           rect.left = 0;
                rect.right = GetDeviceCaps(hdc, HORZRES);
        rect.bottom = GetDeviceCaps(hdc, VERTRES);

      SetMapMode(hdc, MM_ANISOTROPIC);
      SetWindowExt(hdc, rect.right, rect.bottom);
      SetViewportExt(hdc, rect.right, rect.bottom);

      hlf = LocalAlloc(LHND, sizeof(LOGFONT));
      if (hlf == NULL) return;

      plf = (LOGFONT *) LocalLock(hlf);
      if (plf == NULL) return;

      plf->lfHeight = -50;
      lstrcpy(plf->lfFaceName, "Courier New");

      hFont = CreateFontIndirect(plf);

      LocalUnlock(hlf);
      LocalFree(hlf);

      hPtr = (HGLOBAL) GetWindowLong(hWnd, 4);
      if (hPtr == NULL)
         {
         DeleteDC(hdc);
         return;
         }

      ReportStruct = (REPORTSTRUCT *) GlobalLock(hPtr);
      string = (char (*)[PAGE_WIDTH + 5]) GlobalLock(ReportStruct->hReportText);

      ClearMessages();

      hOldCursor = SetCursor(LoadCursor(NULL, IDC_WAIT));

      for (index = 0, lines = 0 ; index < ReportStruct->LineCount; index++)
          {
            if (! lines)
             {
             StartPage(hdc);

             hOldFont = SelectObject(hdc, hFont);
             SetTextAlign(hdc, TA_LEFT);
             GetTextMetrics(hdc, &tm);

             MaxLines = rect.bottom / tm.tmHeight;

             if (Page > 1)
                {
```

```
            memset(HeaderString, ' ', PAGE_WIDTH);
            AtSay(HeaderString,    0, "Year");
            AtSay(HeaderString,    9, "Refuse In Place (Mg)");
            AtSay(HeaderString,   41, "(Mg/yr)");
            AtSay(HeaderString, 57, "(Cubic Ft/yr)");

            TextOut(hdc, 0, lines++ * (tm.tmHeight + tm.tmExternalLeading),
                    HeaderString, strlen(HeaderString));
            memset(HeaderString, '=', PAGE_WIDTH);
            TextOut(hdc, 0, lines++ * (tm.tmHeight + tm.tmExternalLeading),
                    HeaderString, strlen(HeaderString));
            }

          }

        TextOut(hdc, 0, lines * (tm.tmHeight + tm.tmExternalLeading),
                string[index], strlen(string[index]));


          if (++lines >= MaxLines)
            {
            SelectObject(hdc, hOldFont);
            lines = 0;
            Page++;
            EndPage(hdc);
            }
        }

      if (lines)
         {
         SelectObject(hdc, hOldFont);
         EndPage(hdc);
         }

              EndDoc(hdc);
       DeleteDC(hdc);

      DeleteObject(hFont);

      GlobalUnlock(ReportStruct->hReportText);
      GlobalUnlock(hPtr);

      SetCursor(hOldCursor);
      }
}
```

```
// ========================================================================
//                                UTILITY.C
// ========================================================================
// ========================================================================
//                          Miscellaneous functions
// ========================================================================
// ========================================================================
//                            Revision History
// ========================================================================
//  09/30/94    Radian(RLB)             Initial Revision
// ========================================================================

#include <windows.h>


void CenterWindow(HWND hWndCenter)
{
HWND hParent;
HDC hDC;
RECT ChildRect, ParentRect;
int x_place, y_place;
int ParentSizeX, ChildSizeX;
int ParentSizeY, ChildSizeY;

   hParent = GetParent(hWndCenter);

   GetWindowRect(hWndCenter, &ChildRect);
   GetWindowRect(hParent, &ParentRect);

   ParentSizeX = ParentRect.right - ParentRect.left - 1;
   ChildSizeX = ChildRect.right - ChildRect.left - 1;

   ParentSizeY = ParentRect.bottom - ParentRect.top - 1;
   ChildSizeY = ChildRect.bottom - ChildRect.top - 1;

   if (((ParentSizeX - ChildSizeX) < 0) || ((ParentSizeY - ChildSizeY) < 0))
      {
      hDC = GetDC(hParent);
      ParentSizeX = GetDeviceCaps(hDC, HORZRES);
      ParentSizeY = GetDeviceCaps(hDC, VERTRES);
      ReleaseDC(hParent, hDC);

      ParentRect.left = 0;
      ParentRect.top  = 0;
      }

   x_place = ParentRect.left + (ParentSizeX - ChildSizeX) / 2;
   y_place = ParentRect.top + (ParentSizeY - ChildSizeY) / 2;

   MoveWindow(hWndCenter, x_place, y_place, ChildSizeX, ChildSizeY,
              FALSE);
}
```

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

#include "landwin.h"
#include "str_ops.h"


static void ValidationError(char *Msg);


BOOL ValidateData()
{
HWND ActiveWindow;
HGLOBAL hOperData = NULL;
OPERDATASTRUCT *OperData;
double *Period;
BOOL ValidateOPPARAMETERS(OPPARAMETERS *Parameters);
BOOL ValidatePERIODLIST(PERIODLIST *Periods);

   ActiveWindow = SendMessage(RetrieveClientWindow(), WM_MDIGETACTIVE,
                              0, 0);
   if (ActiveWindow != NULL)
      hOperData = SendMessage(ActiveWindow, AM_GETOPERHANDLE, 0, 0);

   if (hOperData != NULL)
      OperData = (OPERDATASTRUCT *) GlobalLock(hOperData);

   if (OperData == NULL) return FALSE;

   if (! ValidateOPPARAMETERS(&(OperData->Parameters))) return FALSE;
   if (! ValidatePERIODLIST(&(OperData->Periods))) return FALSE;
//   if (! ValidateMODELPARAMETERS(&(OperData->Model))) return FALSE;

   return TRUE;
}



BOOL ValidateOPPARAMETERS(OPPARAMETERS *Parameters)
{
   if (Parameters->YearOpened == 0)
      {
      ValidationError("Opening Year must not be zero");
      return FALSE;
      }

   if (Parameters->YearCurrent == 0)
      {
      ValidationError("Current Year must not be zero");
      return FALSE;
      }

   if (Parameters->Capacity == 0)
      {
      ValidationError("Capacity cannot be zero");
      return FALSE;
      }

   return TRUE;
}


BOOL ValidatePERIODLIST(PERIODLIST *Periods)
{
```



```
BOOL NonZero = FALSE;
int ret;
double *Period;
int PeriodCount;
int i;
int Invalid = FALSE;

   PeriodCount = Periods->PeriodCount;
   Period = (double *) GlobalLock(Periods->hPeriod);

// Checks to see that all of the years are positive and the total waste is
// increasing or remaining constant

   for (i = 0 ; i < PeriodCount ; i++)
      {
      if (i > 0)
         {
         if (Period[i] < Period[i - 1]) Invalid = TRUE;
         if (Period[i] > 0) NonZero = TRUE;
         }
      else if (Period[0] < 0) Invalid = TRUE;
      }

   GlobalUnlock(Periods->hPeriod);

   if (Invalid)
      {
      ValidationError("Refuse in place cannot decrease in time");
      return FALSE;
      }

   if (! NonZero)
      {
      ValidationError("Refuse in place must be non zero during operation");
      return FALSE;
      }

   return TRUE;
}



static void ValidationError(char *Msg)
{
   MessageBox(NULL, Msg, "Validation Error", MB_ICONSTOP | MB_OK);
}
```

Eastern Research Group, Inc.

MEMORANDUM

Environmental Science and Engineering

Economic and Regulatory Analysis

Environmental and Occupational Health Services

Software Applications Development

Pollution Prevention

Technology Evaluation

Environmental Sampling and Analysis

Technical Writing and Editing

Conference Management

Graphic Design and Media Services

Public Relations and Outreach

Training and Education

TO: Susan Thorneloe, U.S. EPA

FROM: Chad White, ERG CW

DATE: October 28, 1996

SUBJECT: Draft Help Screen Upgrades for the Windows® and DOS versions of the Landfill Air Emissions Estimation Model; Work Assignment 196.

The enclosed package contains two diskettes: one is an installation disk for the Windows® Version 1.0, Beta Release 4 of the Landfill Air Emissions Estimation Model. The other is an installation disk for the DOS Version 2.0 of the model. These installation disks contain the completed drafts changes to the Help Screens for both the Windows® and DOS versions of the software. A draft version of the user's manual was sent to EPA on October 9, 1996.

Included with this memo as attachment 1 is a status report on the action item list of upgrades that was distributed and discussed at the August 9 meeting between Susan Thorneloe and ERG. All of the recommended upgrades have been addressed.

Included with this memo as attachment 2 is the punch list of suggested upgrades for future project funds. This list includes both software changes and further modifications to the software help screens that were discovered during our study but which were not within the scope of the tasks completed under this work assignment. However, none of these items are critical to fix. These items fall into a category of making the model more user-friendly and easier to implement.

Included with this memo as attachment 3 is a list of the QA/QC activities that were conducted to ensure that all tasks were completed as planned.

---

cc: Martha Smith, U.S. EPA
Clint Burklin, ERG
Project File 96-04

Equal Opportunity Employer
Printed on Recycled Paper

P.O. Box 2010
1600 Perimeter Park
Morrisville, NC 27560-2010
Phone 919-461-1419
Fax 919-461-1418

Corporate Headquarters:
110 Hartwell Avenue
Lexington, MA 02173-3134
Phone 617-674-7200
Fax 617-674-2851

2200 Wilson Boulevard
Suite 400
Arlington, VA 22201-3901
Phone 703-841-0500
Fax 703-841-1440

P.O. Box 2010
900 Perimeter Park
Morrisville, NC 27560-2010
Phone 919-461-1419
Fax 919-461-1579

2455 Horsepen Road
Suite 220
Herndon, VA 20171-3426
Phone 703-713-6500
Fax 703-713-6540

ATTACHMENT 1:

Status Report on Upgrades to the User's Manual, Windows Version Help Screens, and DOS Version Help Screens That Had Been Discussed with EPA on August 9, 1996.

Already Corrected Items (from 6/7/96 Email from S. Thorneloe)
On June 7, 1996, Susan Thorneloe presented a list of items requiring attention; (the list has been attached at the back of this memo). From this list, software processing issues have been addressed (per Bob Bass). Consequently, the following items should be closed and require no further action:

1, 3, 4, 6, 8, 11, 12, 15, 16, 17, 18.

User's Manual Upgrades Completed in the October 9th Draft Manual

1. Describe how to forecast emissions by forward-dating the (Email Item #5)*.
2. Note that the file name appears on the printout for each run (Email Item #7).
3. Describe how the "closure date" is calculated by the computer by default and how the user specification of it automatically changes the waste acceptance rates and refuse accumulated for years with no data (Email Item #9).
4. Describe to user how to change the font size, etc. for printing reports (Email Item #10).
5. Explain the cut, copy, and paste functions (partially completed in section 4.2.3 on page 33) (Email Item #14).
6. Describe the *Autocalc* function; it serves to copy/calculate acceptance rates or landfill in place; explain the sophistication of this function -- ability to create step-wise period in life of the landfill.
7. Describe how to compensate for non-biodegradable debris (subtraction from total if reg. agency agrees) or waste that is in the radius of influence of emissions controls (e.g., a gas extraction system). Care will be given in this case NOT to interpret the rule, but only to instruct the user how to use the software once deductible waste has been identified. (Email Additional Items #1 and #2)
8. State the required pattern of entry; that is, the user must enter capacity before acceptance rate, etc. (partially completed in section 4.2.1 on page 28)
9. Explain that waste accepted does not accumulate until the next year (partially completed in section 4.2.2 of the guide, #5 on page 31).
10. Explain that "closure year" and "current year" cannot be the same in this model. The program will automatically assume that the closure year is the year at which the landfill reaches capacity. If the user chooses to specify the closure year, (s)he should always choose a closure year that is at least year+1 from the current year to avoid a floating point errror. Be sure to explain how the model anticipates/expects the waste at the closure year. Be sure to describe that user's ability to leave years blank and let the computer calculate the needed acceptance rates to reach capacity (ie., closure year by definition is the year at which the landfill reached capacity).
11. Expand table of contents
12. Repaginate table of contents.

* The email item denoted here refers to the June 7, 1996, email from Susan Thorneloe.

Windows Version Help Screen Upgrades Completed and Submitted to EPA on October 28, 1996

1. Describe how to forecast emissions by forward-dating the model (Email Item #5).
2. Note that the file name appears on the printout for each run (Email Item #7).
3. Describe how the "closure date" is calculated by the computer by default and how the user specification of it automatically changes the waste acceptance rates and refuse accumulated for years with no data (Email Item #9).
   Describe to user how to change the font size, etc. for printing reports (Email Item #10).
4. Explain the cut, copy, and paste functions (add text in help screens to explain how these can function properly) (Email Item #14).
5. Describe *Autocalc* macro to copy/calculate acceptance rates or landfill in place; explain the sophistication of this function -- ability to create step-wise period in life of the landfill.
6. Describe how to compensate for non-biodegradable debris (subtraction from total if reg. agency agrees) or waste that is in the radius of influence of emissions controls (e.g., a gas extraction system). Care will be given in this case NOT to interpret the rule, but only to instruct the user how to use the software once deductible waste has been identified. (Email Additional Items #1 and #2)
7. Describe how the user (if he has data available) may enter more HAPs into the program (not an Email Item).
8. Explain that waste accepted does not accumulate until the next year
9. State the required pattern of entry; that is, the user must enter capacity before acceptance rate, etc.
10. Explain that "closure year" and "current year" cannot be the same in this model. The program will automatically assume that the closure year is the year at which the landfill reaches capacity. If the user chooses to specify the closure year, (s)he should always choose a closure year that is at least year+1 from the current year to avoid a floating point errror. Be sure to explain how the model anticipates/expects the waste at the closure year. Be sure to describe that user's ability to leave years blank and let the computer calculate the needed acceptance rates to reach capacity (ie., closure year by definition is the year at which the landfill reached capacity).

DOS Version Help Screen Upgrades and Submitted to EPA on October 28, 1996

1. Describe how to forecast emissions by forward-dating the model (Email Item #5).

   This change was made.

2. Note that the file name appears on the printout for each run (Email Item #7).

   Listing this item as a needed change was an error. This function does not exist in the DOS version of the software.

3. Describe how the "closure date" is calculated by the computer by default and how the user specification of it automatically changes the waste acceptance rates and refuse accumulated for years with no data (Email Item #9).

   This change was made.

4. State the required pattern of entry (i.e., the user must enter capacity before acceptance rate, etc.)

   This change was made.

5. Explain that waste accepted does not accumulate until the next year.

   Listing this item as a needed change was an error. This function does not exist in the DOS version of the software.

6. Describe how to compensate for non-biodegradable debris (subtraction from total if reg. agency agrees) or waste that is in the radius of influence of emissions controls (e.g., a gas extraction system). Care will be given in this case NOT to interpret the rule, but only to instruct the user how to use the software once deductible waste has been identified. (Email Additional Items #1 and #2)

   This change was made.

ATTACHMENT 2:

Punch List for Future or for Unused Funds

Punch List for Future Funds

Upgrades for Windows Version

1. Create an average acceptance rates macro (Email Item #13). This is partially taken care of by the Autocalc function. However, a separate macro would make the software more user-friendly.
2. Change cursor movement for data entry to toggle after an entry (i.e., for movement of cursor with <enter>).
3. Change cursor movement to highlight the cell where cursor is (so that next can simply be overwritten instead of having to move the cursor to the cell with the mouse.
4. Put "Closure Year" on toolbar.
5. Develop ability to load output into a spreadsheet.
6. Change format of data entry cells from scientific notation.
7. Change order/format of help screens (from index-based to topic-based)
8. Prompt user for confirmation when saving over a file of same name.
9. *Current Parameters (Active Model Parameters Dialog Box)*: [k] and [Lo] should be written next to their text descriptions.
10. *Current Parameters* item should be in the *Parameters* menu instead of *Defaults*.
11. *Edit Air Pollutants*: an appended record (i.e., a new HAP entered) should be created blank instead of defaulting with all the information for xylene.
12. When a value for the "current year" and "closure year" are the same, a floating point: division by zero error is encountered. Guidance will be given to instruct the user not to enter equal values for these two variables. Future versions of the software could be changed to correct this problem or provide an error message if they are entered with the same value.
13. Correct Autocalc function so that it will not allow refuse in place to accrue past the landfill capacity; (this phenomenon was observed by choosing a first year acceptance rate equal to half the landfill capacity, highlighting the acceptance rate cells down the column, and then choosing the Autocalc function. A report could not be generated for a trial in which the refuse in place exceeded the capacity of the landfill).
14. Add a function that will clear (delete contents) from a group of cells in the data entry area.
15. Adjust the fonts so that the 32-point font need not be selected to have a 10-point font appear in the printout.

Upgrades for Windows Version (continued)

16. Correct the AP-42 defaults NMOC value -- this value should be different for codisposal and no codisposal. The value is different when a study is opened and initially specified with the opposite landfill types. However, changing the landfill type once a report has already been run fails to make a change in the NMOC concentration.
17. Correct spelling mistake in the AP-42 caution box that appears when AP-42 default parameters are selected: "insure" should be "ensure."

Upgrades for DOS Version

1. Correct spelling mistake: error message at initial menu (when something other than *New Study* is first selected) has a spelling mistake: "edit~~t~~ing" should have only one "t".
2. *Configure Program*: a word should be omitted; the command bar should say "SPACE = Toggle ~~Printer~~ Choices."
3. *Defaults Mode*: "Regulatory" should be changed on the toggle choices to "CAA".
4. *Defaults Mode*: "User-specified" should be on the toggle choices. This value should function as in the Windows version.
5. Error message at initial menu (when something other than *New Study* is first selected) has a spelling mistake: "edit~~t~~ing" should have only one "t".
6. Caution box for AP-42: "~~insure~~" should be "[ensure]".

ATTACHMENT 3:

QA/QC Activities Conducted

QA/QC Activities Conducted on Installable Programs

1. User's Manual upgrades were reviewed by Bob Bass, Rich Pelt, and Clint Burklin for clarity and accuracy. Their recommendations were reflected in the user's manual and help screens.

2. The action list provided to the EPA work assignment manager defining the scope of the suggested upgrades to be made was reviewed. ERG staff ensured that all changes were addressed.

3. Windows and DOS help screen changes were reviewed by Chad White after completion of upgrades by ERG's computer programming staff.

4. A copy of the Windows version of the software with the updated help screens was installed. Installation without problems was assured.

5. The Windows version of software was run. All help screens were used to ensure that all help screens work properly and that changes appear in the correct places. Particularly close attention was paid to the action list of changes listed in the action list provided to the EPA work assignment manager (attachment 1).

6. A copy of the DOS version of the software with the updated help screens was installed. Installation without problems was assured.

7. The DOS version of software was run. All help screens were used to ensure that all help screens work properly and that changes appear in the correct places. Particularly close attention was paid to the action list of changes listed in the action list provided to the EPA work assignment manager (attachment 1).

8. Studies with known results (from the user's manual) were run using the updated installed versions (Windows and DOS) of the software to ensure that the executable files were still fully functional and operating properly.

APPENDIX E

CALCULATION OF
UNCONTROLLED OPEN-TOP VAPOR CLEANER EMISSION FACTOR
INCORPORATING RECYCLE

EXAMPLE CALCULATION: UNCONTROLLED EMISSION FACTOR CONSIDERING RECYCLE

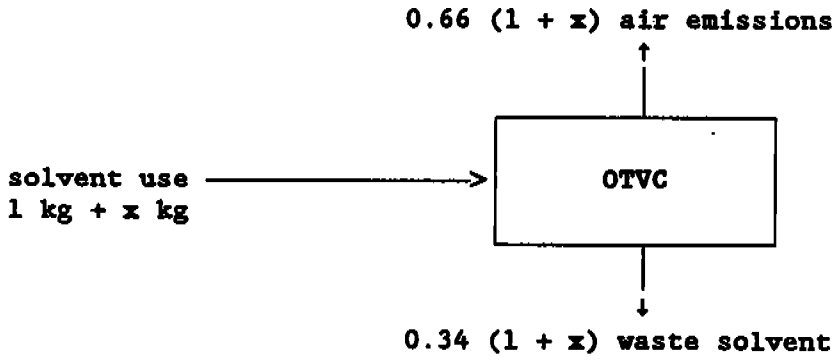
- A material balance is used to estimate emission factors (kg emitted/kg fresh solvent used) considering off-site waste solvent recycle. OTVC factors are shown as an example.

- The emission factor for OTVC not considering waste solvent recycle is 0.66 kg emitted/kg used. The remainder of usage (.34 kg/kg) becomes waste solvent.

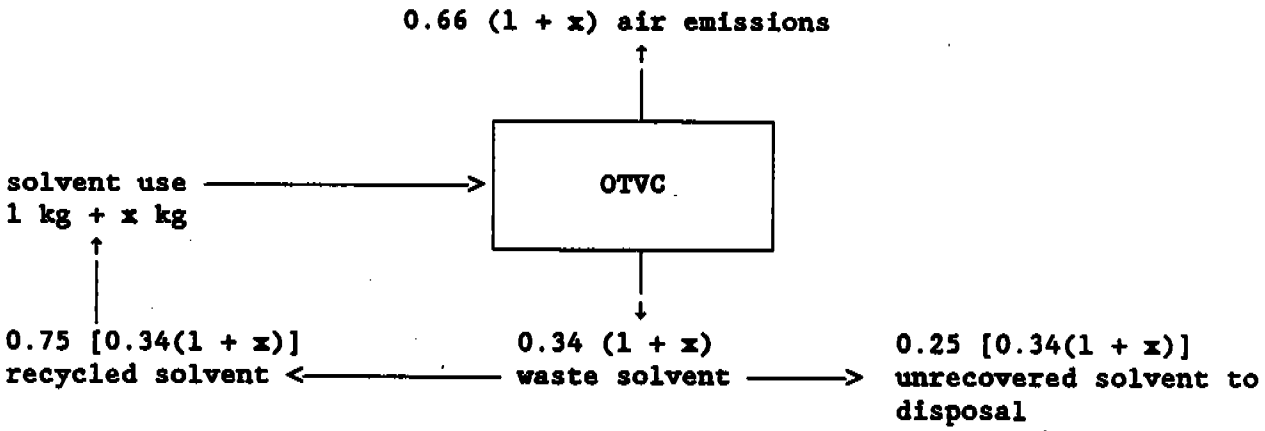
- Let x - recycled solvent use.

- For every 1 kg of fresh (virgin) solvent used:

  - Total solvent use = 1 + x
  - Total air emissions = 0.66 (solvent use) = 0.66 (1 + x)
  - Waste solvent = 0.34 (1 + x)



- It is assumed that 75% of waste solvent is recovered by off-site solvent recyclers and returned for use in cleaning. The other 25% is unrecovered and is disposed of.



- Calculate amount of recycled solvent used

  $x = 0.75\ [0.34(1 + x)]$

  $x = 0.342$
  (kg of recycled solvent used per kg fresh solvent used)

- Calculate air emissions

  $x = 0.66\ (1 + x) = 0.89$
  (kg emitted to the air per kg fresh solvent)

- Thus, emission factor for OTVC considering recycle = 0.89 kg/kg fresh solvent use.

Source: Memorandum from R. C. Mead and R. F. Pandullo, Radian Corporation, to D. Beck, U.S. Environmental Protection Agency. Calculation of Number of Organic Solvent Cleaners and Solvent Emissions and Use Per Model Plant. September 8, 1987.

REFERENCES FOR PLASTICS MANUFACTURING

1. Occupational Safety and Health Administration Proposal for New Methylene Chloride Standard, 56 FR 57047, November 7, 1991.

2. Survey of Methylene Chloride Emission Sources, EPA-450/3-85-015, U.S. Environmental Protection Agency, Research Triangle Park, NC, June 1985.

FLEXIBLE URETHANE FOAM PRODUCTION INDUSTRY

Polyurethane products are generally complex plastics that form from a reaction of liquid isocyanate components with liquid polyol resins.[1] The resin component can also contain blowing agents, combustion retarding agents, and catalysts. Polyurethane products include polyurethane foams, flexible polyurethane foams, and polyurethane elastoplastics.[1] Polyurethane foams are solid.

Methylene chloride is the leading auxiliary foam-blowing agent used in the production of flexible urethane foams. Its use in the foam industry is largely in the production of flexible slabstock foam.[2] It also has some use in the production of flexible molded foam.[3] The development of new catalysts enabled the use of MC in a variety of foam formulations.

Methylene chloride is considered to be a physical blowing agent (also known as "solvent") that assists in foam cell formulation, as it is a low boiling point (39.8°C) halogenated hydrocarbon that does not decompose.[3,4] There are indications that MC is also used to clean the molding and the foam mixing head, and as a carrier solvent for the mold release agent.[3]

There are an estimated 180 foam-blowing companies in the United States, including slabstock and flexible molded foam companies.[1] In 1991, these companies consumed approximately 14 percent of the total MC production.[2] Polyurethane foam industry consumption of MC in 1984 was estimated to be 70 percent slabstock urethane foam and 30 percent flexible molded foam, derived from Halogenated Solvents Industrial Alliance (HSIA) data and Section 114 questionnaires.[3] The 1991 estimate shows a 20 percent increase in use in polyurethane foam production from the 1984 estimate. Flexible urethane molded foam facilities were not located at that time because this product was considered to be a less significant source of MC emissions. The present location of all existing foam-blowing facilities was not determined. The following sections will discuss the industry production processes, emission sources, and emission estimates of MC from flexible urethane foam production.

Process Description

In order to produce a foam, it is necessary to generate nucleating bubbles within a gelling mixture. Polyurethane elastoplastics are produced using either polyether polyols or polyester and diisocyanates (combustion-retarding agents may also be employed). These foams are available in pourable or injectable liquid, preformed pelletized solids, and sheetstock. Flexible foams are produced from polyether polyols, toluene diisocyanate (TDI), and polymeric isocyanates.[1] These foams are generally low-density, soft foams that incorporate carbon dioxide gas as the primary blowing agent.[1] In the production of rigid polyurethane foam (made from polyether polyols, combustion-retarding agents, polymeric isocyanates, and low-boiling halocarbon blowing agents), MC is not used as a blowing agent, but is often employed for filling and cleaning the mixing head.

As mentioned previously, polyurethane foam production (especially slabstock urethane foam) is the primary MC consumption source within polyurethane foam-blowing production, and will be the focus of the discussion in the following text.

Flexible polyurethane foam slabstock and flexible molded foam are produced by the exothermic reaction of TDI with a polyol. As mentioned, carbon dioxide is the primary blowing agent, with MC being used as an auxiliary blowing agent. Production involves the mixing of TDI and polyol directly with the blowing agents, catalysts (i.e., tertiary amines), foam stabilizers, and flame retardants.

The foam-producing reaction occurs within the "foam tunnel" of the process production line. These foam tunnels are generally conveyorized in-line systems enclosed on the sides by plastic curtains. Chemical ingredients are normally pumped from tanks or 55-gallon drums to a mixing head and discharge nozzle.[3] The nozzle pumps the liquid reactants onto the conveyor belt within the tunnel. An exothermic reaction of the chemicals produces the primary blowing agent (e.g., carbon dioxide), which results from the reaction of isocyanate with water, and vaporizes the secondary blowing agent (e.g., MC), producing the foam cells during its formation. The heat

evolved from the reaction of the isocyanate with the polyol and with water is more than sufficient to boil or evaporate MC.

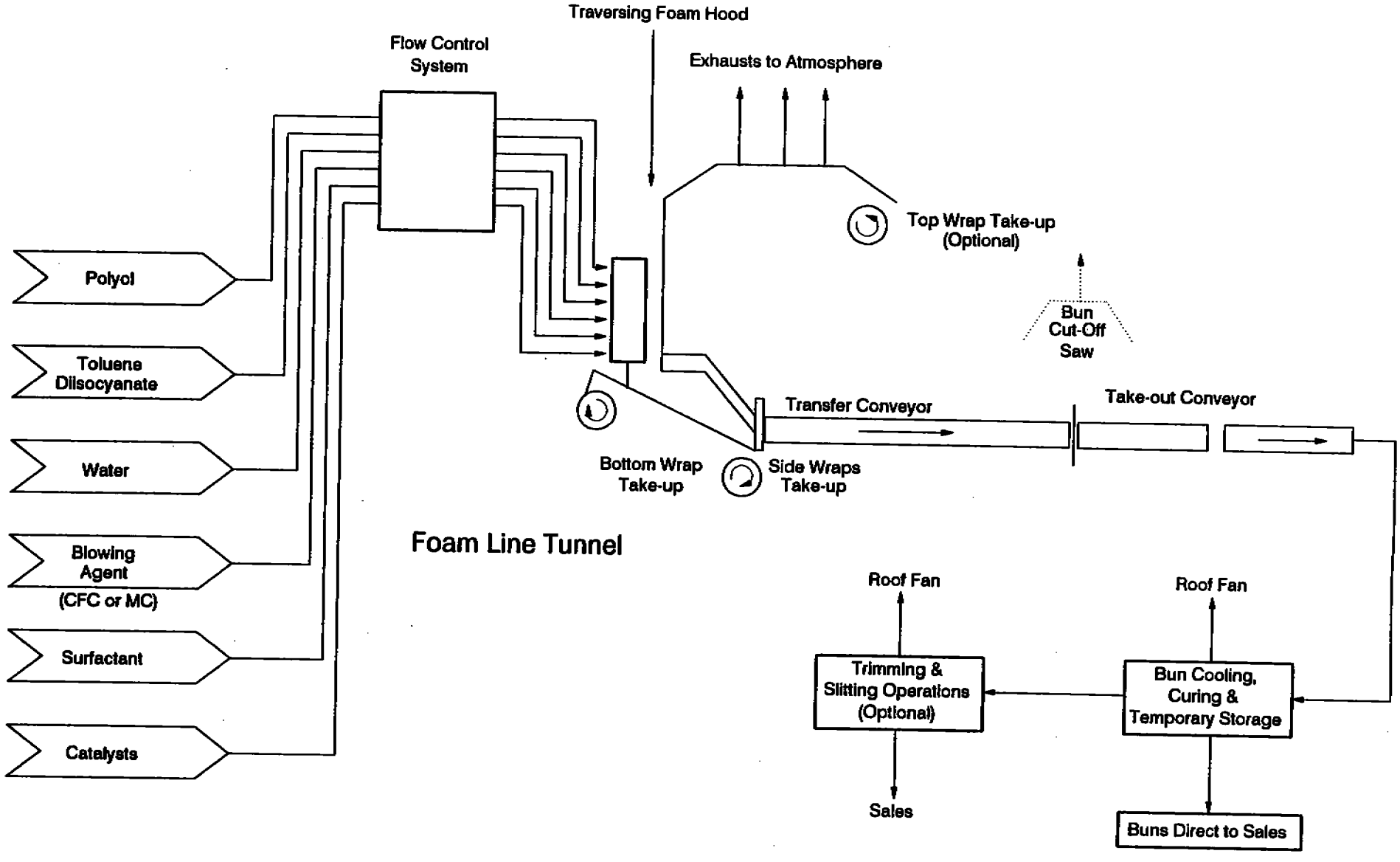
Foam slabs at varying dimensions reach their maximum height within 4 minutes after the chemical liquid reactants are discharged onto the conveyor. When this process involves the reaction mix being poured into a closed mold, it produces a flexible polyurethane molded foam.[1] When such molds are not used, slabstock results. Polymerization (e.g., gelling) reactions and further solidifying of the foam occurs prior to the foam exiting the tunnel (an estimated 10 minutes). Following exit from the tunnel, the foam is further cooled, prepared (e.g., sawed into slabs) and packaged.[3]

One of the most important processing parameters is temperature. Temperature changes can affect the viscosity of the mixture, which influences the pump's metering ability. Pumps are metered to enable proper mixture composition, and differ according to whether high or low pressure machines are used, or whether the process is done on a batch or continuous basis.[1] Figure 6 is a typical schematic flow diagram of flexible polyurethane slabstock foam production that illustrates the foam line tunnel conveyor and product preparation steps.[3]

Emission Sources and Controls

The primary MC emission sources from polyurethane foam production facilities include process vents, equipment leaks, and storage tanks.

Process vent emissions are primarily from vents above the foam tunnel and in the foam curing area. Data obtained from a foam manufacturer in 1986 included mass balance data that indicated that approximately 60 percent of the initial MC charge is emitted in the tunnel and approximately 40 percent is emitted in the curing area.[3] Section 114 questionnaire responses from foam manufacturers reporting the use of MC as an auxiliary blowing agent indicated that control devices were not being used to reduce process vent emissions.[3] Industry still reports that process vent controls are not employed because of the expense; however, technological research on process vent controls is underway.[5] Industry also reports significant research on process modifications to eliminate the use of MC in polyurethane foam production.[5]



Source: Reference 3.

Figure 6. Schematic flow diagram of polyurethane flexible foam production.

An emission factor for uncontrolled process vents (foam tunnel, curing area) in the polyurethane foam (flexible slabstock) production is presented in Table 20.[6] This emission factor was derived on the assumption that MC consumption by the industry equals process vent emissions, plus equipment leak and storage emissions.[3]

Equipment leaks in polyurethane flexible foam production process occur when the liquid or gas process stream leaks from components. The following types of process components are used in foam production: pumps, flanges, liquid valves, gas pressure relief devices, sampling connections, and open-ended lines. As with MC production, the methodologies outlined in the document, "Protocols for Generating Unit-Specific Emission Estimates for Equipment Leaks of VOC and VHAP," can be used to estimate emissions from the production process equipment leaks.[7] An example of one of the simpler methods is presented in Appendix A. An emission factor for uncontrolled MC equipment leaks based on Section 114 questionnaire responses in 1985 is presented in Table 20.[6]

Storage tank emissions can be derived by using EPA Publication No. AP-42 factors with site-specific information.[8] An example calculation is presented in Appendix A. Accidental spills and resulting emissions are considered to be minimal. An estimated storage tank emission factor derived for uncontrolled storage tanks (including fixed-roof tanks and pressurized tanks) using average storage tank data from Section 114 questionnaire responses in 1985 is presented in Table 20.[6] Facilities that have pressurized tanks were assumed to have no emissions.

An aggregate emission factor for the entire production process is also presented in Table 20,[6] and is based on the assumption that all the MC consumed during the process is emitted to the air at some point in the process.

Emissions Control--

Potential control techniques to reduce MC emissions from polyurethane flexible foam production processes, and their estimated control efficiencies are presented in Table 21.[3]

TABLE 20. UNCONTROLLED EMISSION FACTORS FOR POLYURETHANE FOAM PRODUCTION

Emission Source	Emission Factor g/kg (lb/ton) MC Consumed
Process vents (foam tunnel, curing area)	980 (1960)
Equipment leak emissions	17 (34)
Storage tank emission	3 (6)
Entire Process	1000 (2000)

Source: Reference 6.

TABLE 21. CONTROL TECHNIQUES AND EFFICIENCIES USED TO ESTIMATE CONTROLLED EMISSIONS FROM POLYURETHANE FLEXIBLE FOAM PRODUCTION

Emission Source	Control Technique	Percent Reduction in Methylene Chloride Emissions
Process Vents:		
Foam Tunnel	Foam Tunnel Enclosure/ Carbon Adsorption	95[a]
Curing Area	None	0
Storage Tanks	Condenser	85
Equipment Leaks		60-100[b]
Pump Seals		
Packed	Monthly LDAR	60.8
Mechanical	Monthly LDAR	60.8
Valves		
Gas	Monthly LDAR	73
Liquid	Monthly LDAR	59
Pressure Relief Devices		
Gas	Rupture Disk	100
Sample Connections	Closed Purge Sampling	100
Open-Ended Lines	Caps on Open Ends	100

Source: Reference 3.

[a] Assumes 100 percent capture efficiency within foam tunnel.

[b] Depends on control technique for given equipment component.

LDAR = Leak Detection and Repair

As previously discussed, not all production facilities have been located and contacted recently to discern whether controls are more stringent now than in 1985. It is known, however, that MC use within this industry has increased by 20 percent since 1984 as it has increasingly been used as a substitute for CFC-11 in the production process.

REFERENCES FOR POLYURETHANE FOAM INDUSTRY

1. Occupational Safety and Health Administration Proposal for New Methylene Chloride Standard, 56 FR 57045-57046, 57056-57057, November 7, 1991.

2. "Chemical Profile--Methylene Chloride," Chemical Marketing Reporter, 241(9):42, March 2, 1992.

3. "Methylene Chloride Emissions from Flexible Slabstock Polyurethane Foam Facilities," Memorandum from R. L. Ajax and S. R. Wyatt, U.S. Environmental Protection Agency, Radian Corporation, to J. Farmer, Standards Development Branch, U.S. Environmental Protection Agency, Research Triangle Park, NC, April 21, 1986.

4. The Flexible Polyurethane Foam Handbook, Dow Chemical, Urethanes Unlimited, 1985.

5. "Status of Control of Process Vent Emissions," Telephone communication from Joanne O'Loughlin, Radian Corporation, Research Triangle Park, NC, with H. Stone, General Foam Corporation, West Hazelton, PA, April 20, 1992.

6. "Crosswalk/Air Toxic Emission Factor Data Base Management System (XATEF), Version 1.2," Diskette, U.S. Environmental Protection Agency, Research Triangle Park, NC, 1991.

7. Protocols for Generating Unit-Specific Emission Estimates for Equipment Leaks of VOC and HAP, EPA-450/3-88-010. U.S. Environmental Protection Agency, Research Triangle Park, NC, October 1988.

8. "4.3 Storage of Organic Liquids," Compilation of Air Pollutant Emission Factors, Volume I: Stationary Point and Area Sources, Fourth Edition, AP-42, Chapter 12, Supplement E, U.S. Environmental Protection Agency, Research Triangle Park, NC, September 1985.

PHARMACEUTICAL MANUFACTURING

The pharmaceutical manufacturing industry used approximately 11 percent of the total MC consumed in the United States in 1991.[1] Methylene chloride is used in pharmaceutical manufacturing as a general solvent, as an extraction solvent, and in tablet coatings.[2,3] Although most of the MC is used in pill coatings, it is also used in the manufacture of antibiotics, vitamins, contraceptives, and drugs used to control hypertension and diabetes. Many facilities have been able to reduce or eliminate MC from tablet coating operations and substitute water or other safer chemicals. Previous EPA studies indicate that there are over 800 pharmaceutical plants in the United States and its territories,[2] but MC is used in only 76 of these facilities.[4] Table 22 contains a partial list of pharmaceutical manufacturing facilities that use MC. It should be noted, however, that the information is based on a 1985 survey.[3] A survey of 1989 TRIS data revealed 74 pharmaceutical facilities using MC. Refer to Appendix D for a list of these facilities. Ten of the facilities on the TRIS list are found in Table 22.

Methylene chloride is useful in pharmaceutical extractions for a number of reasons. Because of its low boiling point (40°C), it can be used to extract heat-sensitive materials. It is useful in extractions from water because it is immiscible with water, tends not to emulsify, and has a high specific gravity (1.33 at 20°C).[5,6] Some pharmaceutical companies use MC as an extraction solvent because their product is very soluble in MC.[5]

The main reason that MC is used to spray coating on tablets is that it is highly volatile and so evaporates readily. Methylene chloride is also useful if the tablet is sensitive to water and/or heat.[7] Methylene chloride forms a binary azeotrope with water (98.5% by weight at 38°C) and can be used as a drying medium.[5,6] This azeotropic property can be important for coating crystals with another water-soluble solid. A typical coating solution does not consist of just MC, but is also composed of lesser percentages of alcohol and solids. Methylene chloride is completely miscible with other chlorinated solvents, diethyl ether, and ethanol, so the above mixture can be varied to give the best coating.[5,7] Some companies use MC because it dissolves cellulose acetate, which can be used for semi-permeable membranes.[5] Another reason for its prevalent use is that it is easy to control emissions through

TABLE 22. PARTIAL LIST OF PHARMACEUTICAL MANUFACTURING FACILITIES THAT USE METHYLENE CHLORIDE

Facility	Location	Annual Capacity kg/yr (lb/yr)
Abbott Labs[a]	Barceloneta, PR	[b]
	North Chicago, IL	
Aldrich Chemical	Milwaukee, WI	[b]
Beecham, Inc.[a]	Piscataway, NJ	[b]
Biocraft Labs[a]	Waldwick, NJ	2,300,000 (5,000,000)
Bristol-Myers[a]	Syracuse, NY	[b]
Burroughs Wellcome[a]	Greenville, NC	[b]
Chemical Dynamics	S. Plainfield, NJ	270 (600)
Chemical Service	West Chester, PA	45 (100)
Ciba Geigy[a]	Ardsley, NY	[b]
	Summit, NJ	[b]
Deepwater, Inc.	Compton, CA	45 (100)
Eli Lilly & Co.[a]	Indianapolis, IN	[b]
Frank Enterprises	Columbus, OH	[b]
Ganes Chemicals, Inc.	Pennsville, NJ	[b]
Genzyme	Boston, MA	[b]
Henkel of America	Kankakee, IL	[b]
Nepera, Inc.	Harriman, NY	[b]
Pfizer[a]	Groton, CT	[b]
	Terre Haute, IN	
Squibb Corp.[a]	Kenly, NC	10,000 (22,000)
Upjohn[a] (Fine Chemical Div)	Arecibo, PR	[b]
	Kalamazoo, MI	
Warner Lambert	Holland, MI	[b]
William H. Rorer	Fort Washington, PA	[b]

Source: Reference 3.

[a]Also found in TRIS data.

[b]Capacity not available.

NOTE: These operating plants and locations were current as of November 1985. The reader should verify the existence of particular facilities by consulting current listings and/or the plants themselves. The level of MC emissions from any given facility is a function of variables such as capacity, throughput, and control measures, and should be determined through direct contact with plant personnel.

activated carbon absorption, and the MC retained can be reused without further purification.[7]

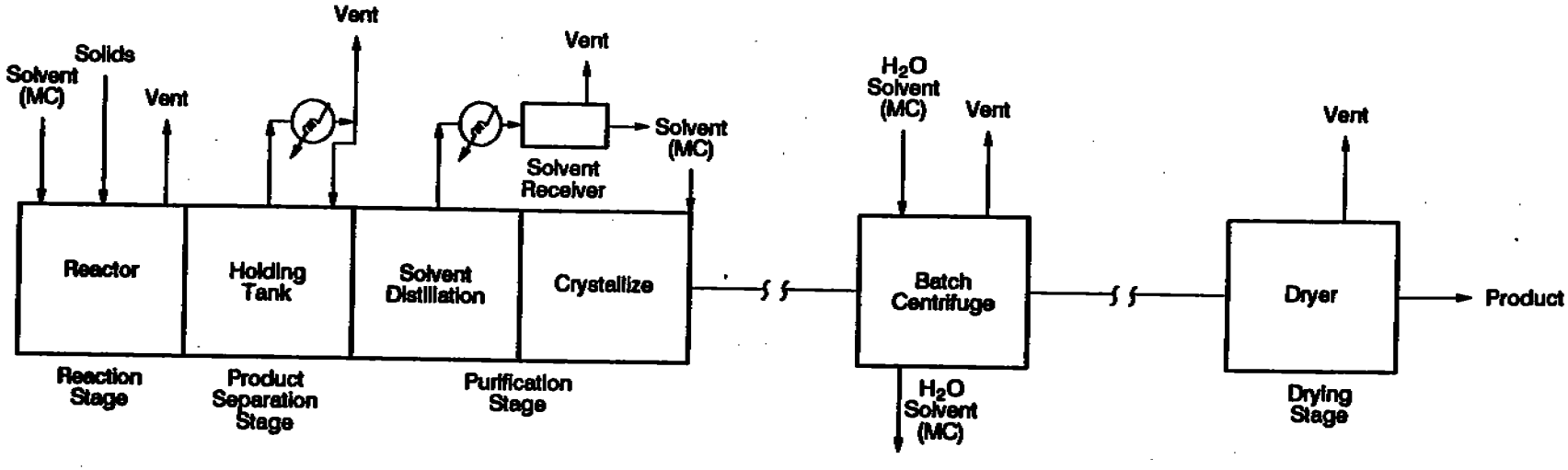
Even though MC has properties conducive to manufacturing pharmaceuticals, there has been an effort to reduce the amount used by the industry because of possible negative health effects. Solvent substitutes such as methanol and ethanol have been considered. However, these substances are not always suitable because of flammability and health concerns. Petroleum distillates and aqueous solutions are being substituted for MC at some facilities.[4,8] Pharmaceutical manufacturing operations are very diverse with some plants using chemical synthesis to produce active ingredients (fermentation and natural extraction are alternative means) and some plants formulating final products (capsules, tablets, etc.). Facilities may conduct one or more of the above operations.

Process Descriptions

Synthetic Organic Pharmaceutical Chemical Process--

Pharmaceuticals typically are manufactured in a series of batch operations. The four successive stages of pharmaceutical production include: chemical reaction, product separation, purification, and drying. Figure 7 shows a typical batch synthesis operation.[2] In the chemical reaction stage, raw material solids and solvents such as MC are mixed in a reactor vessel in which the chemical reaction is carried out, sometimes under elevated temperature and pressure. The stainless steel or glass-lined carbon steel reactor vessel is either an open tank or an enclosed vessel, both equipped with an agitator. Peripheral equipment such as condensers, a refrigeration unit, or a vacuum system can be added to allow the reaction to take place at very high or low temperatures and/or pressures. Some reactors are equipped with a condenser for recirculation of the solvent.

After completion of the chemical reaction, the pharmaceutical products are separated during the product separation stage. The effluent is pumped from the reactor to a holding tank where the reaction products are washed to remove unreacted raw materials and byproducts. The washed reaction products are then piped to various separation process tanks. Product separation often



Source: Reference 2

Figure 7. Typical synthetic organic pharmaceutical chemical process.

622719SR

utilizes an extraction process in which a solvent (such as MC) preferentially dissolves one of the reaction products.

Distillation, crystallization, and filtration are among the purification techniques used after product separation or extraction. Following product separation, the crude extracted product is purified by crystallization of the desired compound from a supersaturated solution. A filter press is usually used to separate the concentrate from the solvent. The purified product and remaining solvent are then separated in a centrifuge. The cake may be further washed with water or another solvent to remove impurities before drying.

After the completion of the purification processes, products are moved to dryers, such as tray, rotary, or fluidized bed dryers, which use hot-air circulation or are operated under a vacuum to remove the remaining solvents or water from the product.[4]

Tablet Coating Process--

Tablets are coated in rotating open-ended pans that range from 90 to 150 cm (36 to 60 inches) in diameter. The coating is sprayed on the tablets in the pan while warm air (30°C) flows across the pan at a typical rate of 28 cubic meters per minute (1000 cubic feet per minute). The coating solution is made up of MC and alcohol (about 70/30) but water alone can be used.[8] The air evaporates the solvents, leaving coated tablets. Spray coating and drying takes 2 to 3 hours per batch. A large plant might have 20 pans, whereas a small plant might have only two. Any number of pans can be in use at any given time. The pans are usually cleaned after each batch, even if multiple batches of the same material are made.

Pharmaceutical products may also be coated by the Wurster process. In this process, the tablets or pellets are suspended in a fluidized bed while the spray solution is applied using a stream of heated nitrogen.[7,8] Methylene chloride is used as a solvent along with alcohol (70/30) to dissolve the solids used to coat the tablets. This solution is then sprayed on the cores, the solvents evaporated off, and the vapors condensed and collected in a tank for reuse in the next batch.[8] This method is used most often for coating pellets (smaller particles that are later encapsulated), whereas coating pans

are used most often for coating tablets (standard dosages). A good example of pellet coating would be over-the-counter 12-hour cold capsules.

Most tablets are coated with sugar, methyl cellulose, or ethyl cellulose. Cellulose coatings may use either a water or an organic solvent such as MC. Chloroform can be used in place of MC. The use of water as a solvent or solvent component reduces VOC emissions, but more time and heat are required to evaporate the water than for an organic medium. Therefore, this is a production consideration. Also, products that are sensitive to water and/or heat may preclude the use of aqueous coatings. The use of heat or vacuum can expedite evaporation, but this rapid evaporation can peel or roughen the coating.

According to one manufacturer, the rotating pan units can process batches between 400 and 800 kg (900 and 1,700 lbs), with a total yearly throughout between 192,000 and 363,000 kg (423,000 lbs and 799,000 lbs), of which 71,140 kg to 134,380 kg (156,510 lbs to 295,630 lbs) is product.[8] Batch sizes using the Wurster process may vary, with a minimum of 860 kg/batch (1,891 lbs/batch) to a maximum of 3,787 kg/batch (8,331 lbs/batch). In a year's time, a total of 112,820 kg (248,203 lbs) of material were processed through the Wurster column, of which 12,072 kg (26,558 lbs) was product.[8]

Emission Sources

Pharmaceutical Chemical Synthesis--

Methylene chloride is released during storage, transfer, reaction, separation, purification, and drying processes of pharmaceutical chemical synthesis.[4] Storage emissions result from displacement of air containing the solvent during tank charging. Chemical transfer operations, such as manually pouring solvent drum contents, is a source of emissions. Reactor emissions result from the displacement of air containing MC during reactor charging, solvent evaporation during the reaction cycle, venting of uncondensed MC from the overhead condenser during refluxing, purging of vaporized MC following a solvent wash, and opening of reactors during the reaction cycle to take quality control samples. Distillation condensers can emit MC as uncondensed solvent.

During crystallization, emissions can result from the venting of vaporized solvent if the crystallization is being done by solvent evaporation. If crystallization is accomplished by cooling of the solution, there are few emissions. Dryers are potentially large emission sources; emission rates vary during drying cycles, and with the type of dryer being used. Emissions from air dryers are normally greater than those from vacuum dryers mainly because air dryer emissions are more dilute and difficult to control.[2,4]

Below is a ranking, in order of decreasing emissions, that illustrates relative expected total VOC emissions from uncontrolled pharmaceutical chemical synthesis process sources.[2]

- Dryers
- Reactors
- Distillation systems
- Storage and transfer systems
- Filters
- Extractors
- Centrifuges
- Crystallizers

For most pharmaceutical facilities, the first four listed process sources will account for the great majority of total plant MC emissions.[2] In addition to the eight sources listed above, fugitive emissions result from leaks in equipment components. The list differs if controlled emissions are considered because emissions from reactors and distillation systems can often be very efficiently condensed.

Tablet Coating--

Most emissions from tablet coating are process vent exhaust emissions from pan tablet coating. Although the exhaust emissions are often very dilute, they can be controlled with activated carbon adsorption,[7] which enables the manufacturing facility to recover the MC solvent. Emissions from

tablet coating storage and transfer operations, as well as fugitive emissions, are similar to those from pharmaceutical chemical synthesis.

The Wurster process is operated as a totally closed system with a solvent recovery system based on a refrigerated condenser maintained at about 25°C, so emissions from this process are limited to approximately 2 percent.[8]

Emissions Data and Controls

Emission Factors--

Surveys of drug manufacturers in 1975, 1982, and 1985 estimated the final disposition of total MC usage. The responding firms were estimated to represent approximately one-half of the production of ethical (prescription) domestic pharmaceuticals in those years. The amount emitted into the air (instead of being incinerated, disposed of into the sewer, etc.) varied from 43 to 67 percent of total MC consumed.[9-12] Table 23 illustrates the disposition of MC for all three years.

Some emission factors were developed from 1985 process data obtained from the Ciba-Geigy facility in Summit, New Jersey. This information is summarized in Table 24.[8] Emission factors for tablet coating are shown both prior to control and after control by carbon adsorption. No other current emissions data from pharmaceutical manufacturing facilities were located.

As discussed previously in Section 4 for MC producers, site-specific emissions estimates can be developed using the AP-42 methodologies for storage tanks, the "Protocols" methodologies for equipment leaks, and the wastewater CTC document methodology for wastewater. Example calculations are given in Appendix A.

Applicable Controls for Pharmaceutical Chemical Synthesis--

Applicable controls for the vented emissions mentioned earlier, except storage and transfer, are: condensers, scrubbers, and carbon adsorbers.[2] Incinerators are not currently widely used to control vapor phase organic emissions from synthesized drug production facilities.[2] Part of the lack of

TABLE 23. METHYLENE CHLORIDE PURCHASES AND ULTIMATE DISPOSITION BY PHARMACEUTICAL MANUFACTURERS

Year of Data	Annual Purchase (metric tons)	Ultimate Disposition (percent)					
		Air Emissions	Sewer	Incineration	Solid Waste or Contract Haul	Other Disposal	Product
1975	10,000[a]	53	5	20	22	--	--
1982	11,375[b]	43	5	38	11	3	--
1985	1,539[c]	67	8	4	10	7	3

Source: References 9-12.

[a] Data represent 26 pharmaceutical manufacturers, which account for approximately 53% of 1975 domestic sales of ethical pharmaceuticals.

[b] Data represent 17 pharmaceutical manufacturers, which account for approximately 50% of 1982 domestic sales of ethical pharmaceuticals.

[c] Data represent 13 pharmaceutical manufacturers. Information concerning percentage of domestic sales (as in 1975 and 1982 data) not available.

TABLE 24. METHYLENE CHLORIDE EMISSION FACTORS FOR PHARMACEUTICAL MANUFACTURING

Industrial Process	Emission Source	Emission Factor	Control Status
Pan tablet coating	Process Vents	0.053 kg MC/kg product 0.001 kg MC/kg product	Uncontrolled Controlled (dual carbon bed adsorber)
Blender	Process Vent	0.003 kg MC/kg active ingredient processed	Uncontrolled
Coating solution holding tank	Process Vent	0.01 kg MC/kg coating solution processed	Uncontrolled
Coating solution mixer	Process Vent	0.0001 kg MC/kg coating solution processed	Uncontrolled

Source: Reference 8.

NOTE: Emission data are for one facility only and do not represent average emissions for all such sources, or total emissions for all sources.

use may be due to the variability of waste gases that would be ducted to an incinerator and the batch nature of the processes. Fluctuating flows and pollutant concentrations may hamper safe and efficient operation. Therefore, incinerators would most likely find application where relatively stable waste gas flows can be established. Stability may be enhanced by ducting emissions from several sources to a common control device. It should be noted that incineration of MC results in hydrogen chloride (HCl), which is also an air pollutant.

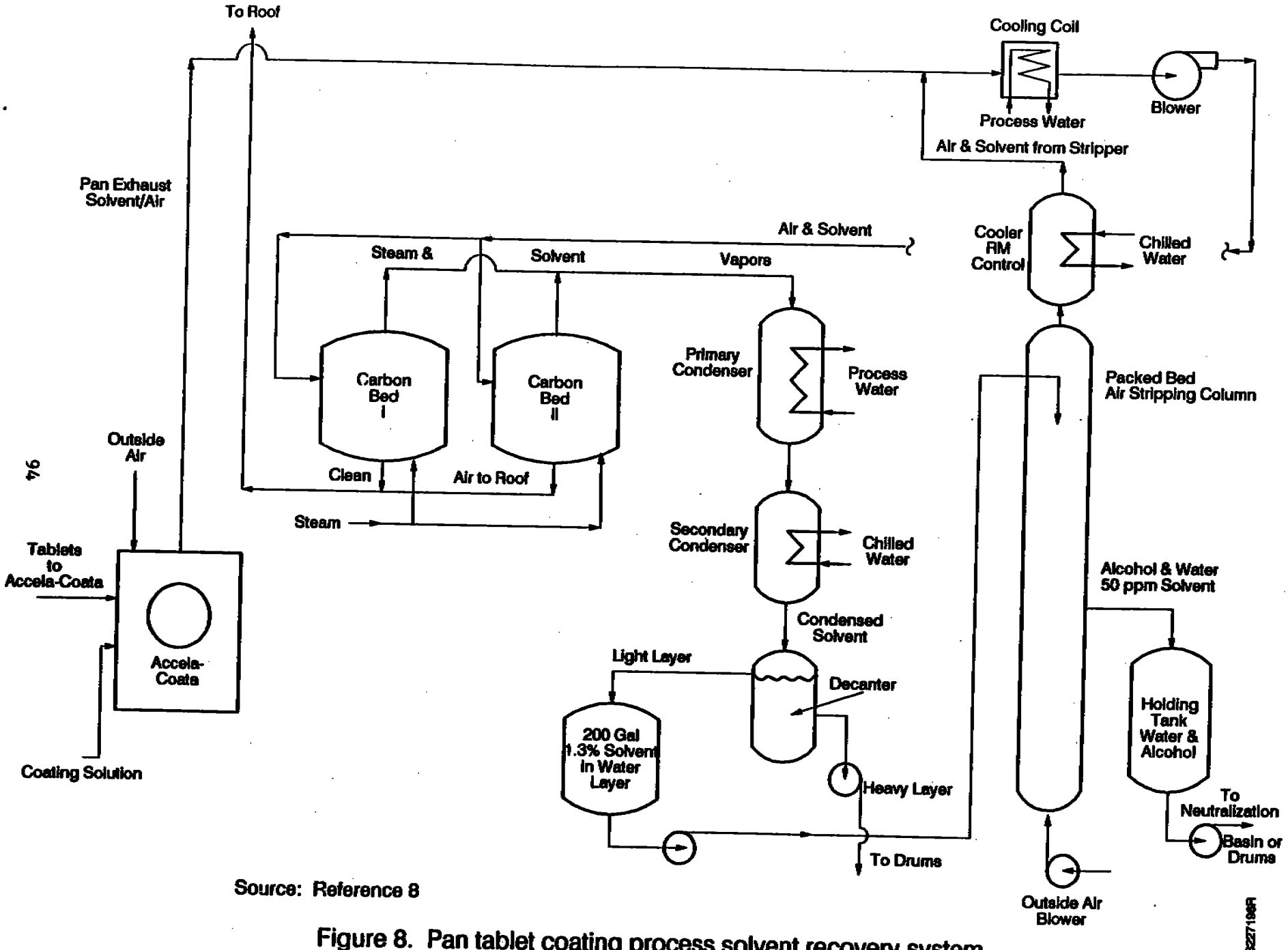
Another potential disadvantage of using incinerators is that heat recovery is likely to be uneconomical because at pharmaceutical plants incinerators will be relatively small and the potential energy recovery correspondingly small, especially when viewed in light of the costs for installing heat recovery equipment.[2] In addition, the incinerator would generally run less than 24 hours a day. In this case, heat recovery would be intermittent, thus decreasing its utility.

Storage emissions can be controlled by storing MC in pressure tanks or by venting storage emissions to a control device such as a condenser, scrubber, carbon adsorber, or combustion device. Floating roofs would be feasible controls for large, vertical storage tanks.[2] These controls are the same as those applicable to emissions from MC production. Transfer may be controlled by vapor balancing, where MC vapors are returned to the storage tanks; or transfer emissions may be vented to a control device.

Control of equipment leak emissions may be accomplished through a regular inspection and maintenance program, as well as by equipment modification. See Section 4 for more information regarding equipment leaks.

Applicable Controls for Tablet Coating--

Figure 8 is a schematic of the pan tablet coating process solvent recovery system.[8] In this control method, the MC-contaminated air from the dryer is passed through a bed of activated carbon (with control efficiencies of 98+ percent). When the carbon bed becomes loaded with organic compounds, it is stripped with low-pressure steam. Because MC is insoluble in water, it is easy to separate from the steam condensate for reuse. Any ethanol that is

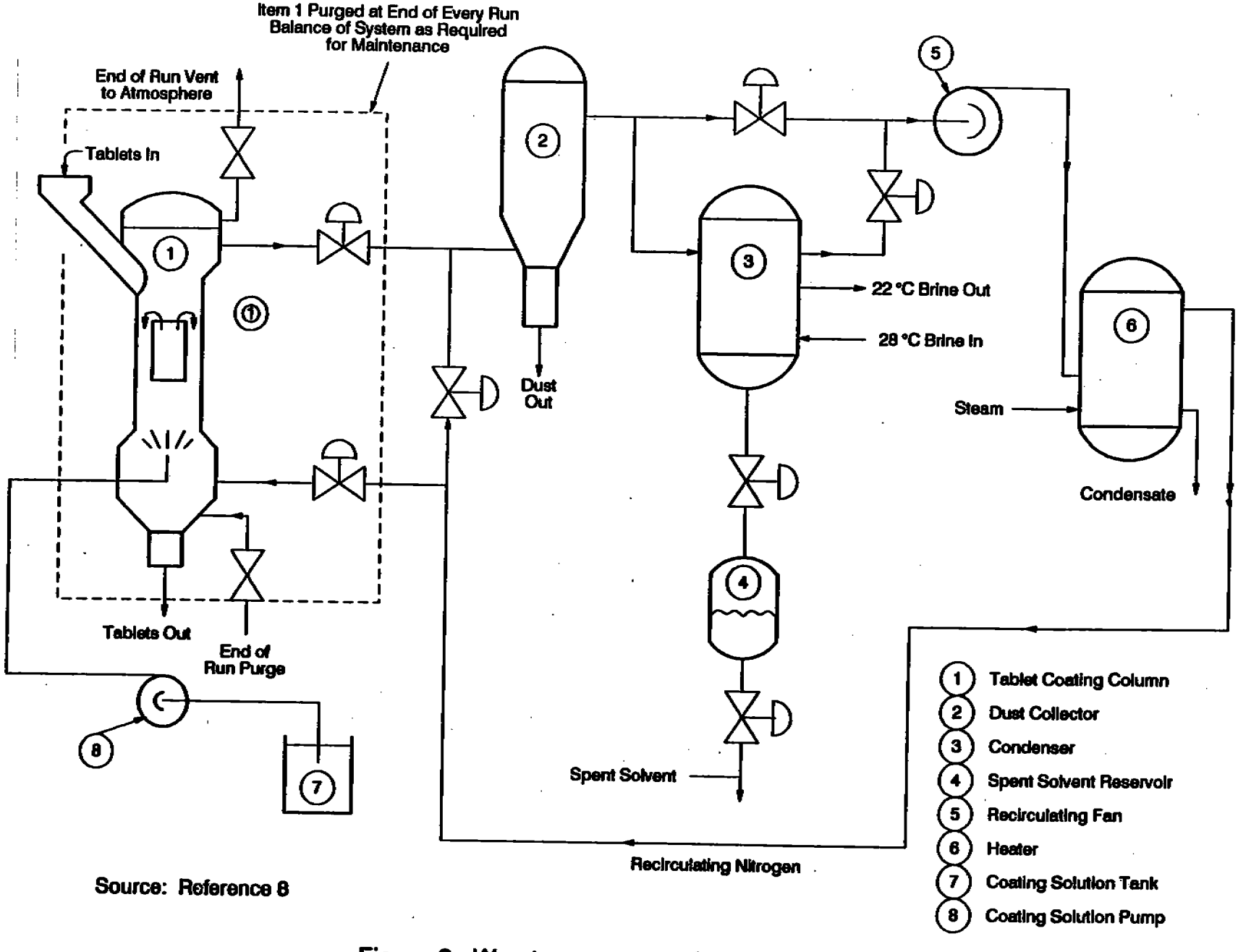


Source: Reference 8

Figure 8. Pan tablet coating process solvent recovery system.

captured is miscible with the steam condensate and is impractical to salvage. The condensate, which contains 1-2 percent alcohol, is usually discarded to a sewer. In a large pharmaceutical plant, this wastewater stream is processed in the plant wastewater treatment system.[7]

Figure 9 describes the Wurster process solvent recovery system.[8] With this system, about 98 percent of the solvents are recovered and reused as is. Because there is no contact with water, no solvents find their way to the sewer system.[8] The remaining 2 percent of solvents are presumably emitted into the air.



Source: Reference 8

Figure 9. Wurster process solvent recovery system.

6227197R

REFERENCES FOR PHARMACEUTICAL MANUFACTURING

1. "Chemical Profile--Methylene Chloride," Chemical Marketing Reporter, 241(9):42, March 2, 1992.

2. Control of Volatile Organic Emissions from Manufacture of Synthesized Pharmaceutical Products, OAQPS Guideline Series, EPA-450/2-78-029, U.S. Environmental Protection Agency, Research Triangle Park, NC, December 1978.

3. "Telephone Survey of Pharmaceutical Manufacturing Industry," Memorandum from R. Nash, Radian Corporation, Research Triangle Park, NC, to D. Beck, U.S. Environmental Protection Agency, Research Triangle Park, NC, November 6, 1985.

4. Occupational Safety and Health Administration Proposal for New Methylene Chloride Standard, 56 FR 57048, 57060, November 7, 1991.

5. "Telephone and Literature Survey of Pharmaceutical Production," Memorandum from N. Cole, Radian Corporation, Research Triangle Park, NC, to D. Beck, U.S. Environmental Protection Agency, Research Triangle Park, NC, February 12, 1986.

6. Kirk-Othmer Encyclopedia of Chemical Technology, M. Grayson, ed., Volume 5, pp. 380, 686, 692. John Wiley & Sons, New York, NY. 1979.

7. Volatile Organic Compound Emission Controls for Tablet Coatings at Pharmaceutical Plants, NTIS No. PB84-182922, U.S. Environmental Protection Agency, Research Triangle Park, NC, January 1984.

8. Written communication from L. DellaVecchia, Ciba-Geigy Facility, Summit, NJ, to Radian Corporation, Research Triangle Park, NC, February 7, 1986.

9. Survey of Methylene Chloride Emission Sources, EPA-450/3-85-015. U.S. Environmental Protection Agency, Research Triangle Park, NC, June 1985.

10. Written communication from T. White, Pharmaceutical Manufacturers Association, Research Triangle Park, NC, to D. Beck, U.S. Environmental Protection Agency, Research Triangle Park, NC, June 8, 1984.

11. Written communication from T. White, Pharmaceutical Manufacturers Association, Research Triangle Park, NC, to D. Beck, U.S. Environmental Protection Agency, Research Triangle Park, NC, August 18, 1986.

12. "4.3 Storage of Organic Liquids," Compilation of Air Pollutant Emission Factors: Volume 1, Stationary Point and Area Sources, Fourth Edition, AP-42, Chapter 12, Supplement E, U.S. Environmental Protection Agency, Research Triangle Park, NC, September 1985.

SOLVENT CLEANING AND PHOTORESIST STRIPPING

An estimated 11 percent (16,420 Mg) of total 1991 United States MC consumption was used for metal cleaning (also called degreasing) in a variety of manufacturing processes, and 3 percent (3,400 Mg) was used for photoresist stripping in printed circuit board manufacture in the electronics industry.[1,2]

Solvent cleaning is a process used to remove water-insoluble soils from metal, plastic, fiberglass, printed circuit boards, and other surfaces. Water-insoluble soils include grease, oil, waxes, carbon deposits, fluxes, tars, metal chips, mold-release agents, and oxidation layers. Solvent cleaning is used by a variety of industries that employ cleaning processes as part of their manufacturing process or prior to painting, plating, inspection, repair, assembly, heat treatment, and machining. Typical industries that use solvent cleaning processes include furniture and fixtures, fabricated metal production, electric and electronic equipment, transportation equipment, plumbing fixtures, aerospace manufacturing, miscellaneous manufacturing, primary metals, automobile and electric tool repair shops, and railroad, bus, aircraft, and truck maintenance facilities.[3] Because of the large number of solvent cleaning operations existing within many different industries, information on the location of the individual solvent cleaning equipment is difficult to obtain. The following sections discuss the solvent cleaning industry, cleaning process descriptions, and emissions.

Process Descriptions

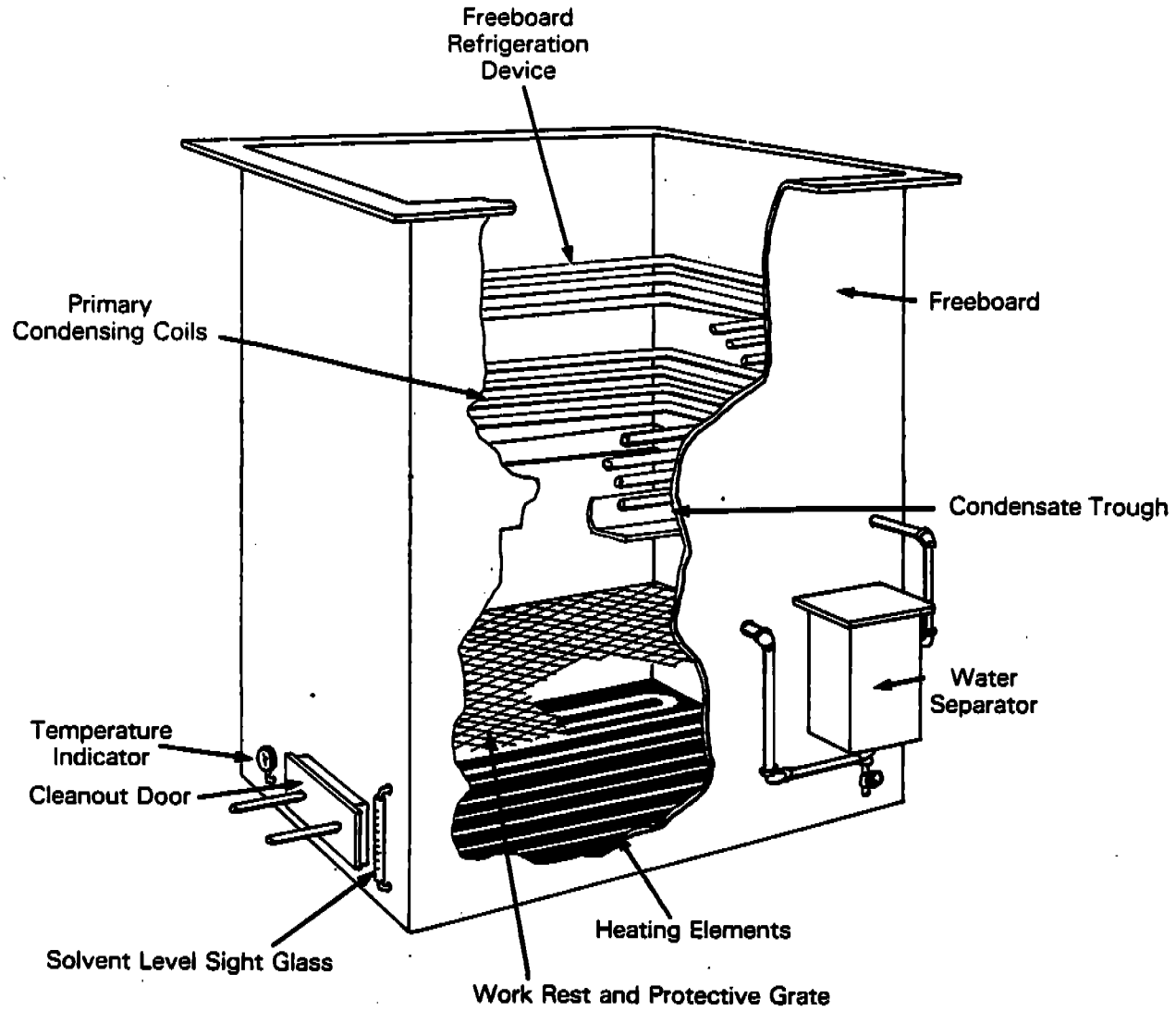
Solvent cleaning processes are typically performed by two basic types of solvent cleaning equipment: batch cleaners and in-line cleaners (also called continuous cleaners). Both cleaners exist in designs for use with solvent at room temperature (cold cleaners) or solvent vapor (vapor cleaners). Methylene chloride solvent cleaning processes for each solvent cleaner [e.g., batch vapor cleaners, in-line cleaners (cold and vapor), and batch cold cleaners] are discussed in the following sections.

Batch Vapor Cleaner Process Description--

Batch vapor cleaners heat solvent to a boiling point, creating a solvent vapor zone into which items to be cleaned are lowered. The cleaning process involves the solvent vapor condensing on the item and stripping soils away. Cleaning can also be supplemented or replaced by immersing items into the liquid solvent during the cleaning cycle. Batch vapor cleaners include open-top vapor cleaners (OTVC), and non-OTVC batch cleaners developed with design variations to meet particular workload characteristics and cleaning demands for particular applications.[3]

Open-top vapor cleaner and non-OTVC batch cleaners are designed to generate and contain solvent vapor. The basic OTVC batch cleaner is illustrated in Figure 10. It is equipped with a heating system or pump to boil liquid solvent. As the solvent boils, dense solvent vapors rise to the level of the primary condensing coils. The primary condensing coils circulate coolant (e.g., water, refrigerant) through the coils, providing continuous condensation of the rising solvent vapors and creating a controlled vapor zone that prevents most vapor from escaping the tank. Solvent vapor and moisture in the air collect in a condensate trough along the sides of the OTVC below the primary condensing coils. This condensate goes into a water separator. The water separator is a container that separates the water from the liquid solvent, returning solvent to the cleaner and routing water for use in another process within the facility/plant or to disposal to a publicly owned treatment works (POTW) system. Some batch OTVC cleaners may also use a canister of desiccant to replace or aid the water separator in its reduction of water contamination. The OTVC walls also extend above the top of the vapor zone. This area is called the freeboard. A freeboard reduces air currents and disturbance of the vapor zone boundary.[3]

Design variations of OTVC batch cleaners are numerous, and depend on the particular characteristics and demands of the workload. Examples of design variations incorporated in OTVC batch cleaners include stills, lip or slot exhausts, covers, and multiple-chamber cleaners. Stills are used to extract soils from the solvent sump and return clean solvent to the machine, decreasing the need to replace the cleaning solvent because of impurities. Lip or slot exhausts are designed to capture solvent vapors escaping from the



Source: Reference 3

Figure 10. Open top vapor cleaner.

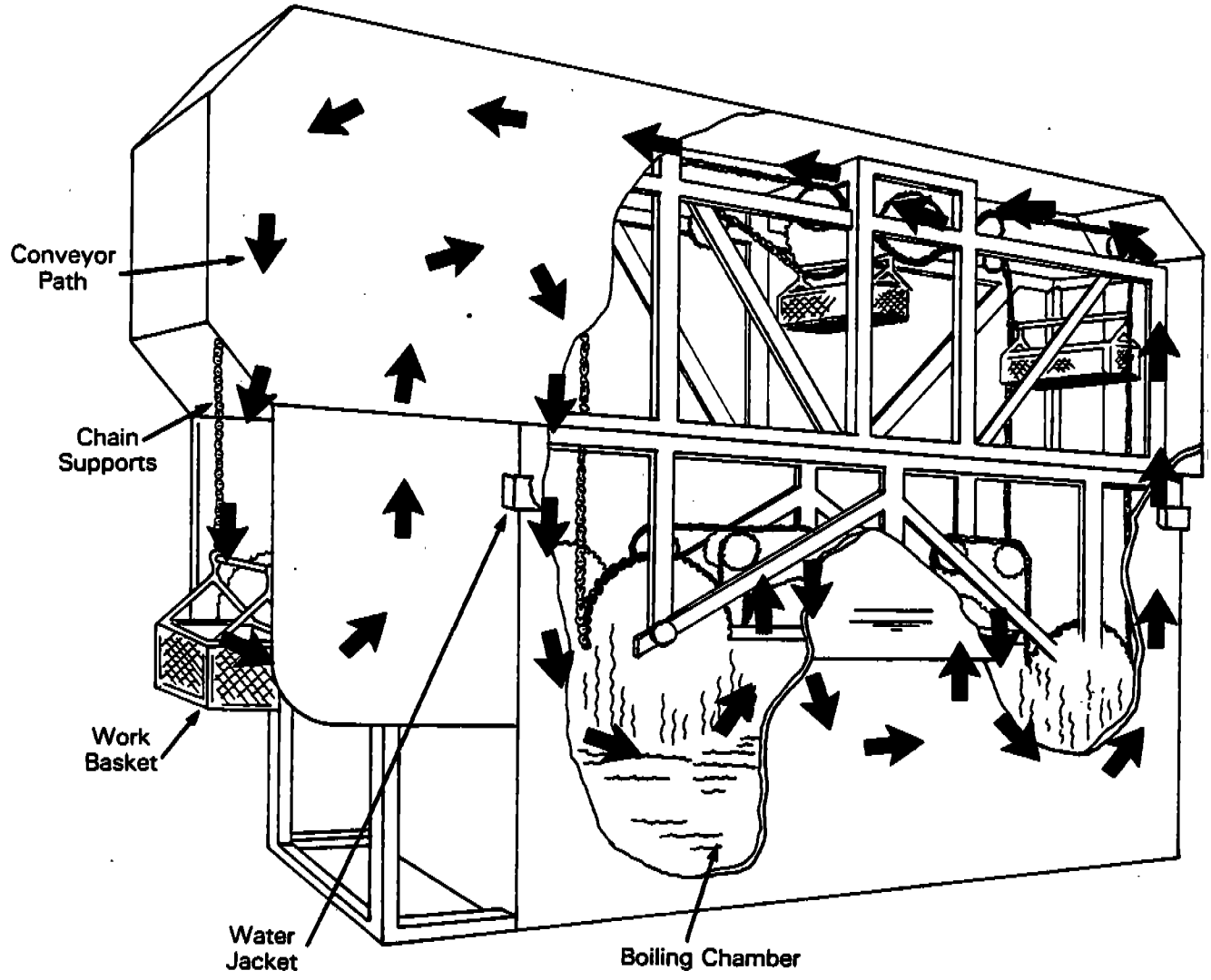
OTVC and carry them away from the work area, and are incorporated to reduce occupational exposure. Covers, in varying designs, are used to limit solvent losses and contamination during downtime or idling time. Multiple-chamber design variations can include various solvent-cleaning methods (e.g., vapor, immersion, spraying).[3]

There are some batch vapor cleaners that are not OTVCs. These include cleaners into which a batch of parts is loaded, and then moved through the cleaner on a conveyor (called a conveyorized batch cleaner) and batch cleaners that are more enclosed than OTVCs. These batch vapor cleaners tend to be larger than the OTVC batch cleaners and employ similar cleaning methods (e.g., condensing vapor, immersion, spray). These cleaners are a hybrid of an OTVC and continuous cleaner. Examples include cross-rod, vibra, ferris wheel, and carousel cleaners. An example of a cross-rod, non-OTVC batch vapor cleaner is illustrated in Figure 11.[3]

The cleaning process for OTVC and non-OTVC batch cleaners entails solvent vapors condensing on the cooler workload entering the vapor zone until the workload temperature approaches the temperature of the vapor. The condensing solvent dissolves and flushes soils from the workload until condensation ceases and the vapor-phase cleaning process is complete. As discussed previously, the vapor cleaning process can also include immersion of the item to be cleaned into the hot, liquid solvent.

Immersion batch vapor cleaning processes often include the use of ultrasonics. Ultrasonics uses high-frequency sound waves that produce pressure waves in the liquid solvent. The areas of low pressure within the solvent form small vapor pockets that collapse as the pressure in the zone cycles to high pressure. The creation and collapse of these vapor pockets aids in cleaning by providing a scrubbing action.

Because of their higher boiling points, impurities (e.g., grease, soil, wax, etc.) from the cleaning process minimally contaminate the solvent vapors. Solvent can be used in vapor cleaning for a longer time than in cold cleaning because of the affinity of the solvent vapor to remain relatively pure as compared to immersion cold cleaner solvent. Another variation in the cleaning process is the use of spray solvent below the vapor line. The pressure of the



Source: Reference 3

1682393R

Figure 11. Cross-rod cleaner.

spray and/or the potential for solvent condensation on the workload aids in the physical cleaning of the workload.

In-Line (Vapor and Nonvapor) Cleaner Process Description--

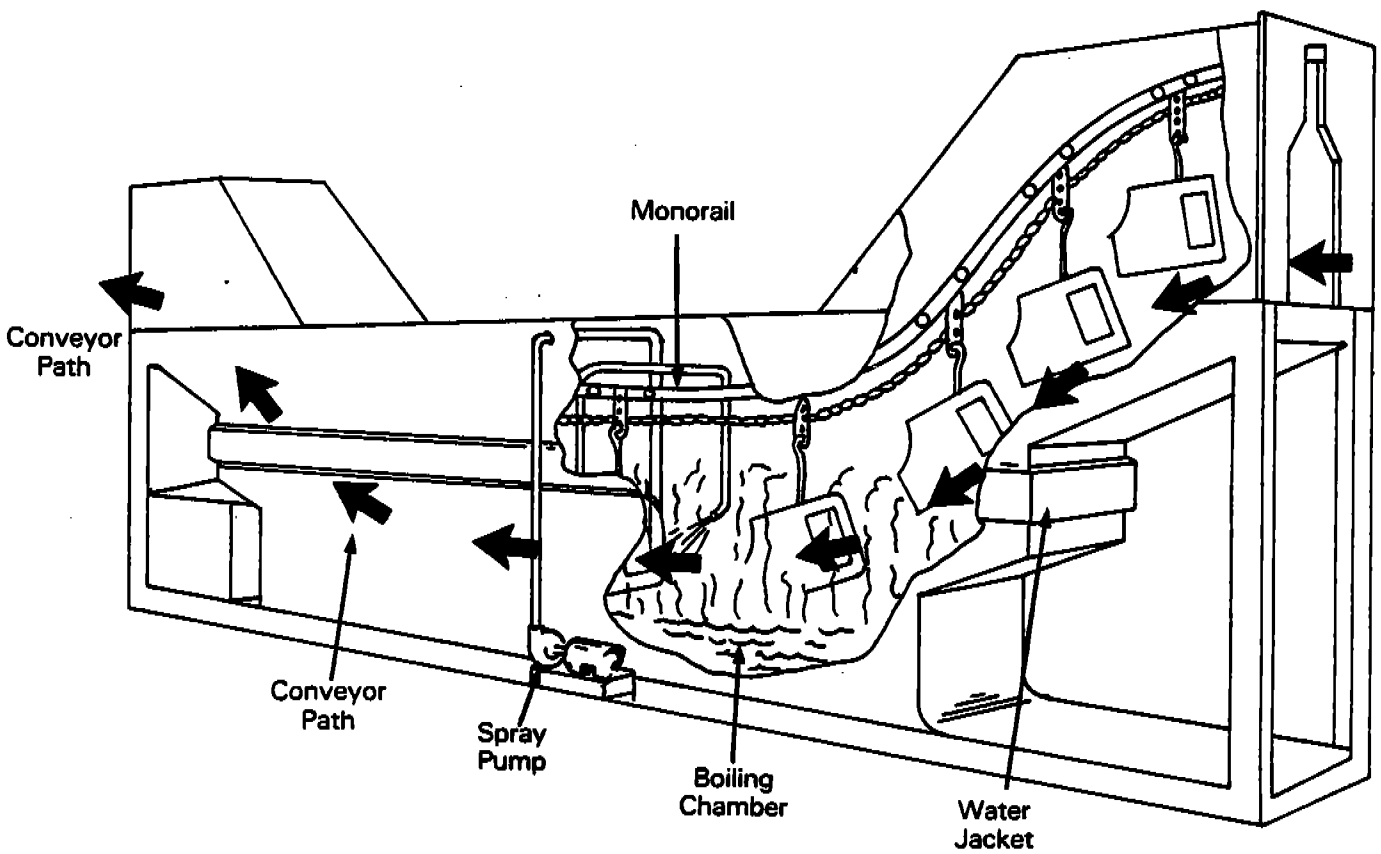
In-line cleaners (also known as continuous cleaners) are cleaners that use automated loading on a continuous basis. The same cleaning techniques are employed in in-line vapor cleaners as with batch vapor cleaners. In-line nonvapor cleaning involves the use of solvent at room temperature, where immersion and spray cleaning techniques can be employed. Most of these cleaners, however, operate as vapor cleaners.[3]

In-line cleaners are usually enclosed, except at the inlet and exit openings where the parts and conveyance pass. They also are typically employed in industries that demand a larger-scale cleaning operation. Design variations within these cleaners are determined by the workload and production rate required. In-line cleaners include monorail, belt, strip, printed circuit board processing equipment (i.e., photoresist strippers, flux cleaners, and developers), and modified cross-rod non-OTVC batch cleaners with both an entry and exit port. An example of a monorail in-line cleaner is illustrated in Figure 12.[3]

Photoresist stripping processes involve using MC to remove any unwanted resist from printed circuit boards. In 1989, 68 percent of MC reported for use in the electronics industry was used in photoresist stripping.[3] Assuming this percentage remained the same for 1991, an estimated 3,400 Mg would have been consumed for use in photoresist operations in 1991.[1,2] A diagram of an in-line photoresist stripping machine is illustrated in Figure 13.[3]

Batch Cold Cleaner Process Description--

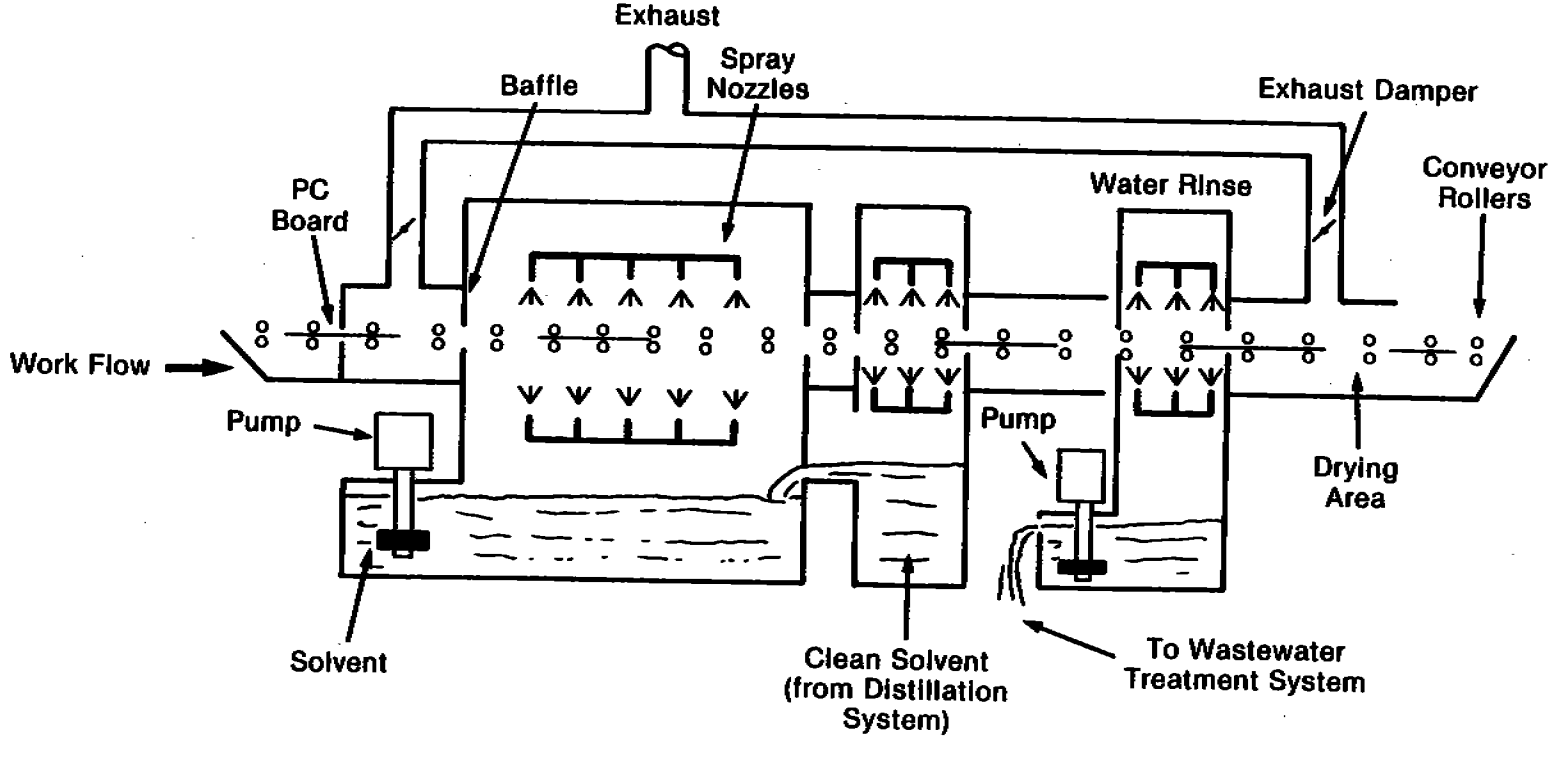
Cold cleaners are usually used in small cleaning solvent maintenance demand situations. The solvent cold cleaning process involves the use of the solvent at room temperature. Cleaning is accomplished by spraying, flushing, wipe cleaning, agitating, or immersing of item to be cleaned with the solvent.



Source: Reference 3

16823394R

Figure 12. Monorail in-line cleaner.



Source: Reference 3

Figure 13. Schematic diagram of an in-line photoresist stripping machine.

The only known machine specifically manufactured for cold cleaning purposes (except for non-vapor in-line cleaners) are carburetor cleaners used in automobile repair operations. Methylene chloride is used in these cleaners to increase the soil dissolving power and reduce the flammability potential of the solvent cleaning blend employed. An example of a carburetor cold cleaner is illustrated in Figure 14.[3]

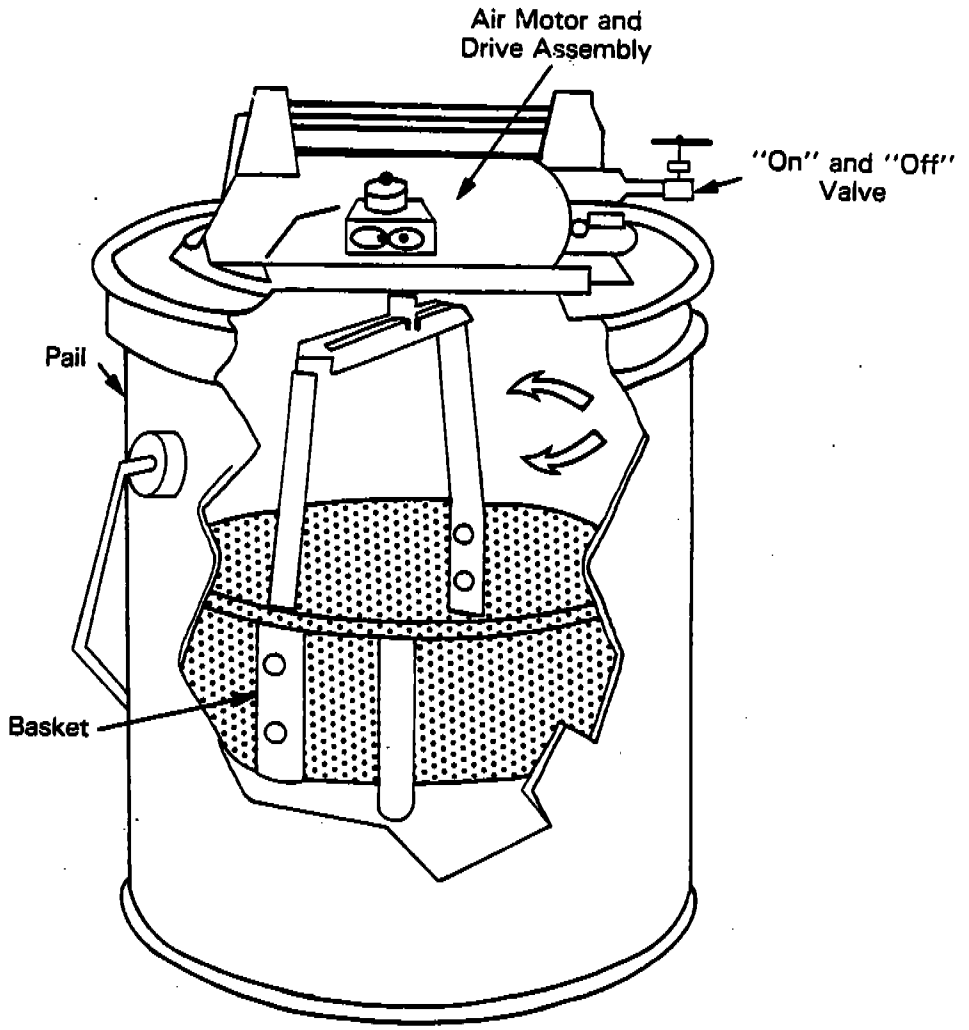
Emissions Sources

Methylene chloride emissions from organic solvent cleaners are air/solvent vapor interface emissions and workload-related emissions. Air/solvent vapor interface emissions that result during idling conditions (when a machine is turned on and ready to operate) are from solvent vapor diffusion and convection. Workload-related emissions result from the introduction and extraction of items cleaned during the cleaning process and spraying processes (if employed) including emissions that occur by solvent carry-out on the workload. Other solvent emission sources include leaks from cleaners or associated equipment, filling and draining operations, and startup, shutdown, and downtime operations.[3]

Idling Solvent Vapor Emissions--

Air/solvent vapor interface emissions under idling conditions in OTVC batch cleaners result mainly from the diffusion of solvent vapors from the vapor zone to the ambient air. Convection losses occur when the heat of a boiling solvent is translated to the solvent cleaner walls, creating a convective upward flow of solvent vapor to the outside of the cleaner. When air flow is introduced across the air/solvent vapor interface because of draft or lip exhaust, the diffusion rate and convection of solvent vapor to ambient air increases. Figure 15 illustrates batch cleaner idling emission sources.[3]

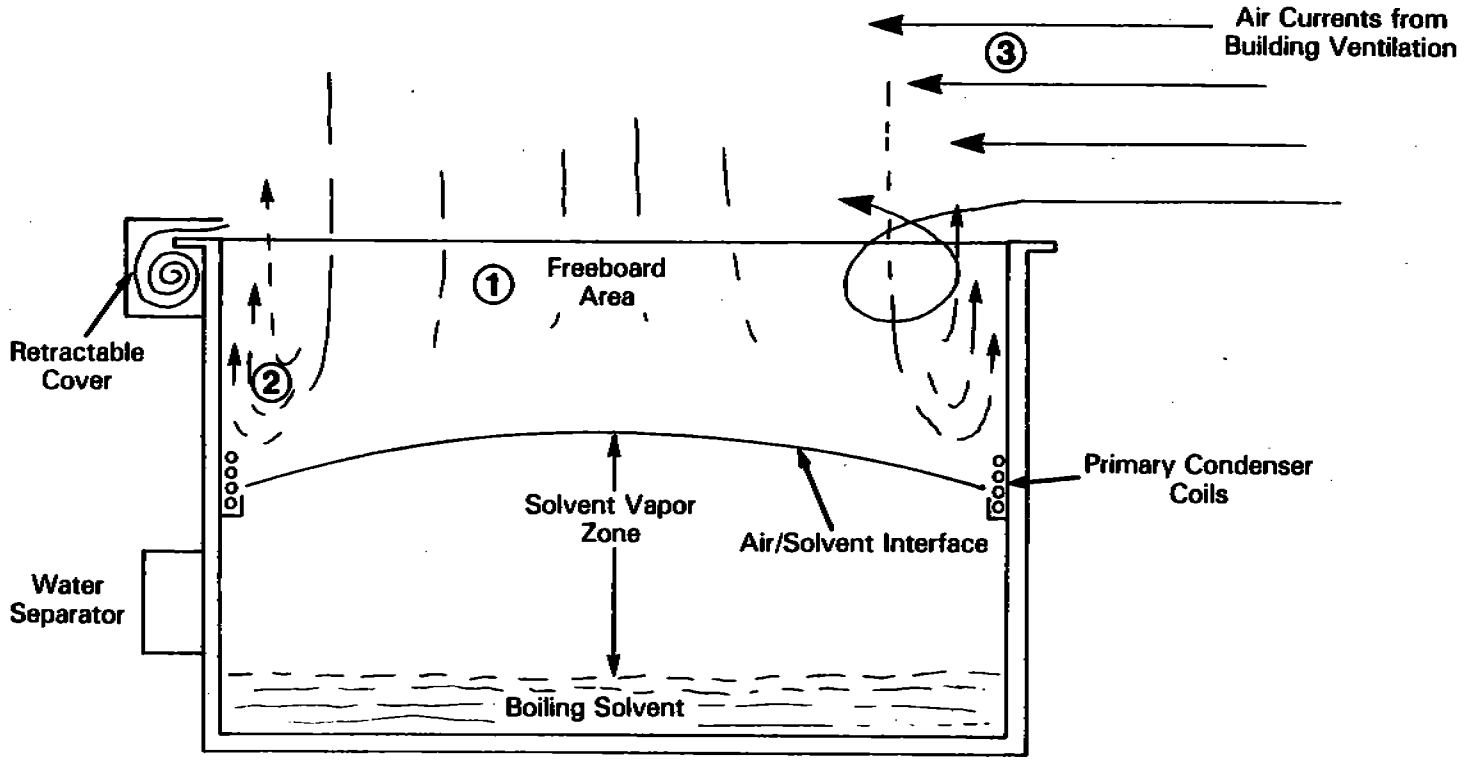
In-line and non-OTVC batch cleaner idling air/solvent vapor loss mechanisms are the same as for OTVC batch cleaners (e.g., diffusion, convection). Figure 16 illustrates these emission sources for an in-line cleaner. The solvent emissions from in-line and non-OTVC cleaners are expected to be less than from OTVC cleaners because these cleaners are more



Source: Reference 3

1794039R

Figure 14. Carburetor cleaner.

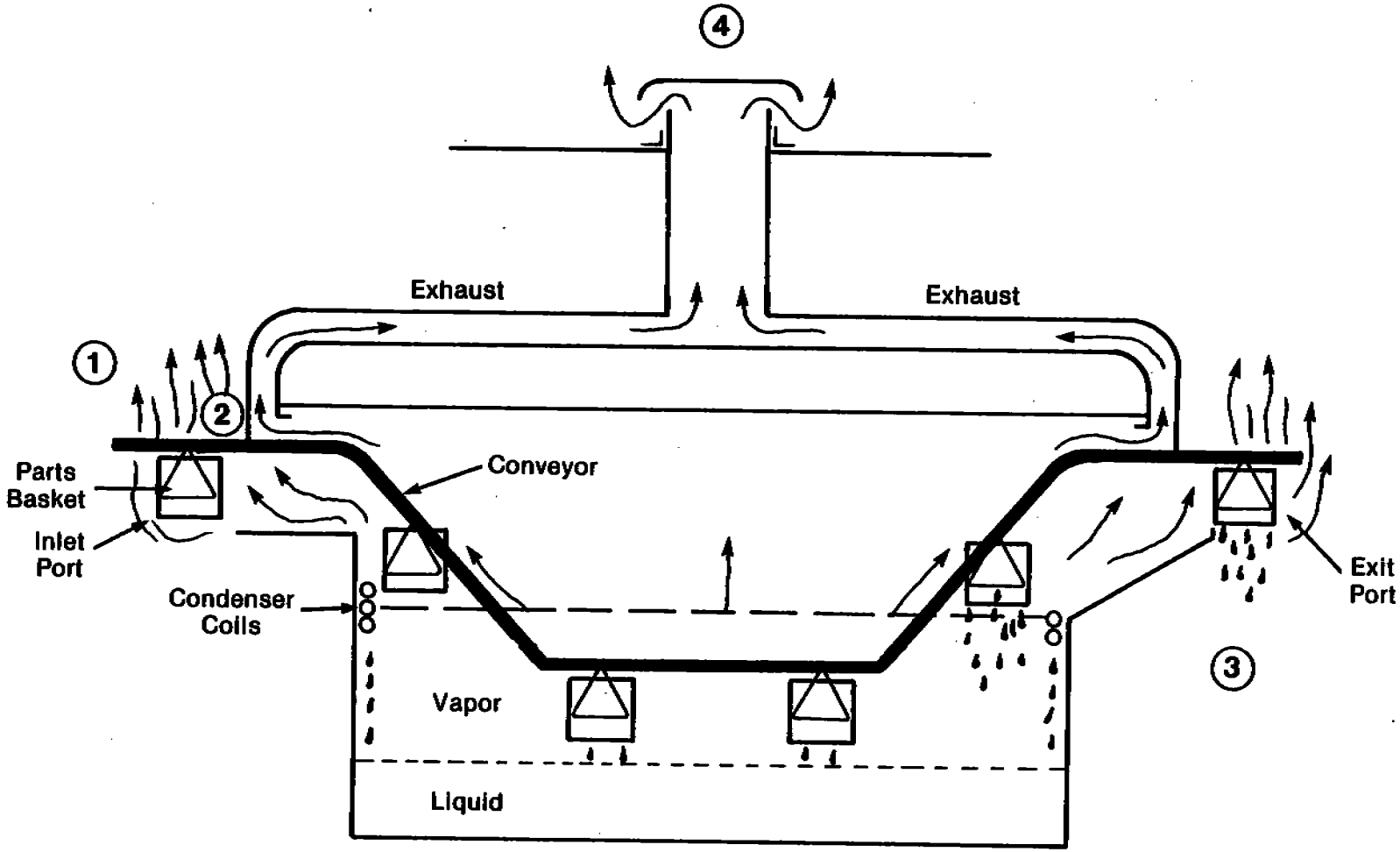


1. Diffusion of Solvent from Air/Solvent Vapor Interface
2. Convection of Solvent Vapor up Warm Tank Walls
3. Diffusion and Convection Emissions Accelerated by Drafts Across Tank Lip (or by Operation of Lip Exhaust Device)

Source: Reference 3

1692415R

Figure 15. Batch cleaner idling emission sources.



1. Diffusion of solvent from air/solvent vapor interface
2. Vapor up warm tank walls
3. Carry-out of liquid solvent on part and subsequent evaporation
4. Roof vent exhaust

Source: Reference 3

0290722R

Figure 16. In-line cleaner emission sources.

enclosed and therefore less exposed to drafts and their associated air/solvent vapor emissions.[3]

Cold cleaner air/solvent vapor emissions under idling conditions occur from evaporation and diffusion. The only known, manufactured cold cleaner is a carburetor cleaner that generally uses MC with an overlaying water layer (MC is heavier than water), so minimal solvent is expected to evaporate.[3]

Workload-Related Solvent Vapor Emissions--

Workload-related solvent losses from OTVC batch cleaners are due to the turbulence and vapor line fluctuation that occur at the air/solvent vapor interface when items to be cleaned enter and exit the vapor zone. Turbulence occurs when the items to be cleaned enter the cleaner. Emissions are influenced by the means of conveyance (e.g., manual or automated). Emissions occur from diffusion and convection, and increase with the speed of transfer of the items into and out of the cleaner.[3] An automated hoist system set at a fairly low and even speed can reduce both in-plant and atmospheric emissions (when emissions are vented to the atmosphere) by reducing the disturbance of the air/solvent vapor interface.

Solvent loss from work-load-related conditions also occurs when solvent spray cleaning is employed. Solvent spray cleaning causes turbulence in the air/solvent vapor interface. Pooled liquid solvent and residual solvent film remaining on the items cleaned after removal from a cleaner can also be a source of solvent emissions to the air. These are called carry-out losses. If a longer dwell time (i.e., length of time the part remains in the vapor zone)[4] and parts orientation to facilitate drainage of pooled solvent is incorporated, liquid solvent carry-out emissions can be decreased.[3] Simple working practices, such as increasing the part dwell time and parts orientation considerations may offer significant emission reductions (reducing in-plant and atmospheric emissions).

In-line and non-OTVC batch cleaner workload-related solvent vapor emissions are similar to emissions from OTVC batch cleaners. Workload-related emissions from these cleaners, however, are less on a per-part basis than those from manually operated OTVCs. Turbulence at the air/solvent vapor

interface (or the air/solvent interface for in-line cold cleaners) is less for these cleaners than for the manually operated OTVCs because of the automation and associated speed control of parts through the cleaning process. Exhaust systems in these cleaners, unless controlled by a carbon adsorber, can result in significant solvent emissions since air movement by exhaust systems may increase diffusion and convection emissions.[3] It is important to note that although exhaust systems may decrease worker exposure in-plant, there is an associated increase in emissions to the atmosphere.

Workload-related solvent emissions from cold cleaners result from solvent agitation and spraying, and solvent liquid and film carry-out. Efforts to facilitate drainage (i.e., tipping of parts, longer drainage time) decrease solvent carry-out emissions.[3]

Other Emission Sources--

Other solvent emissions sources include storage and handling operations, startup, shutdown, and downtime operations, leaks, wastewater, filling and draining operations, distillation operations, and solvent decomposition. These losses will depend on the cleaning machine integrity and design, and the operating techniques employed. Emissions for storage, leaks, and handling losses from the solvent cleaning industry can be estimated by the same methodology as discussed for the MC production industry in Section 4. Appendix A presents an example of simple calculations for fixed-roof storage tanks and equipment leaks; but as described in Section 4, there are also other methods of emission estimation available for equipment leaks and other storage tank configurations. Particular facilities and processes will require differing factors, and reference to EPA Publication No. AP-42 for storage and the "protocols" document for equipment leaks is suggested.

Emission Controls

Solvent control strategies involve machine design and operating practices to minimize emissions from the sources discussed. Available control techniques (including hardware and operating practices) for batch OTVC, in-line (vapor and nonvapor), and cold cleaner operations are shown in Tables 25, 26, and 27, respectively. The EPA published a control techniques guideline

TABLE 25. AVAILABLE CONTROL TECHNIQUES FOR OTVC OPERATIONS

Source of Solvent Loss	Available Control Hardware	Operating Practices
Air/Solvent Vapor Interface	• 1.0 freeboard ratio (FBR) (or higher) • Freeboard refrigeration device • Reduced primary condenser temperature • Automated Cover • Enclosed design • Carbon adsorber • Reduced air/solvent vapor interface area	• Place machine where there are no drafts • Close cover during idle periods
Workload	• Automated parts handling at 3.4 meters per minute (11 fpm) or less • Carbon adsorber • Hot vapor recycle/superheated vapor system	• Rack parts so that solvent drains properly • Conduct spraying at a downward angle and within the vapor zone • Keep workload in vapor zone until condensation ceases • Allow parts to dry within machine freeboard area before removal
Fugitive	• Sump cooling system for downtime • Downtime cover • Closed piping for solvent and waste solvent transfers • Leakproof connections; proper materials of construction for machine parts and gaskets	• Routine leak inspection and maintenance • Close cover during downtime

Source: Reference 3.

TABLE 26. AVAILABLE CONTROL TECHNIQUES FOR IN-LINE OPERATIONS

Source of Solvent Loss	Machine Design	Operating Practices
Air/Solvent Vapor Interface[b]	• 1.0 freeboard ratio • Freeboard refrigeration device[a] • Reduced primary condenser temperature[a] • Carbon adsorber • Minimized openings (clearance between parts and edge of machine opening is less than 10 cm or 10% of the width of the opening)	
Workload	• Conveyor speed at 3.4 meters per minute (11 fpm) or less • Carbon adsorber • Hot vapor recycle/superheated vapor system	• Rack parts so that solvent drains properly • Conduct spraying at a downward angle and within the vapor zone[a] • Keep workload in vapor zone until condensation ceases • Allow parts to dry within machine before removal
Fugitive	• Sump cooling system for downtime • Downtime cover or flaps • Closed piping for solvent and waste solvent transfers • Leakproof connections; proper materials of construction for machine parts and gaskets	• Routine leak inspection and maintenance • Close ports during downtime

Source: Reference 3.

[a] Applies to in-line vapor cleaners, but not in-line cold cleaners.

[b] Air/solvent interface for in-line cold cleaners.

TABLE 27. AVAILABLE CONTROL TECHNIQUES FOR COLD CLEANERS

Machine Design	Operating Practices
• Manual Cover	• Close machine during idling and downtime
• Water cover with internal baffles	• Drain cleaned parts for at least 15 seconds before removal
• Drainage facility (internal)	• Conduct spraying only within the confines of the cleaner

Source: Reference 3.

(CTG) document for solvent metal cleaning in 1977, and an alternative control technology document for halogenated solvent cleaners in 1989. Thirty-three States and the District of Columbia adopted the CTG-based RACT for solvent cleaning emission.

The CTG developed two levels of control (A and B). Control System A specified simple control equipment (e.g., covers and implementation of good operating practices), and System B required that there be other control equipment (i.e., freeboard extension, freeboard refrigeration device) installed in addition to the System A controls.[3] Presently, a proposal for a National Emission Standard for Hazardous Air Pollutants (NESHAP) for the control of halogenated solvent emissions from cleaners is being developed, and a regulation and supporting document is scheduled to be finalized in 1994.

Emission Estimates

Solvent usage and emission factors for uncontrolled and controlled cleaners are shown in Table 28.[4,5] In this table, the uncontrolled emission factors are expressed in two ways. The factors on the first line are expressed in terms of MC emitted per total MC used in the cleaning operation. (The fraction not emitted is contained in waste solvent.) These factors may be more representative for estimating emissions from an individual facility that has information on the total MC it purchased (or consumed) for cleaning, regardless of whether that MC was fresh (virgin) solvent from an MC production plant or MC recovered from waste solvent and re-sold by a solvent recycling company.

The second line of factors were developed for estimating emissions from national data on how much fresh MC produced by MC producers was used for solvent cleaning. These factors are expressed in terms of emissions per kg of fresh MC used (see footnote "c").

National baseline emissions calculated using these emission factors are presented in Table 29.[4] These estimates take into account regulated and non-regulated counties in the United States. To estimate emissions, solvent usage for unregulated counties was multiplied by an emission factor that represented

TABLE 28. METHYLENE CHLORIDE USAGE AND EMISSION FACTORS FOR UNCONTROLLED AND CONTROLLED CLEANERS

Parameter	Cold Cleaners	Carburetor Cleaners[a]	OTVCs	In-line vapor cleaners	Photoresist strippers
Emissions Parameter					
Uncontrolled EF w/o recycle (kg emitted/kg total solvent used (fresh and recycled))[b]	0.66	0.66	0.78	0.85	0.70
Uncontrolled EF w/recycle (kg emitted/kg fresh solvent used)[c,d]	0.89	0.89	0.93	0.96	0.90
CTG Control System B efficiency (%)	17.0	60.0	40.0	60.0	60.0
Controlled EF w/recycle (kg emitted/kg fresh solvent used)[c,d]	0.87	0.76	0.89	0.91	0.78
Relative controlled fresh solvent usage (%)[e]	0.85	0.47	0.63	0.43	0.46

Sources: References 4 and 5.

[a] It is assumed that all carburetor cleaners are controlled at baseline, so only the controlled emission factor is used in calculations.

[b] This is the amount emitted by a cleaner per kg of total solvent (MC purchased for cleaning (includes virgin fresh solvent plus solvent bought from recycling companies).

[c] Emission factors are expressed on a fresh solvent (MC) feed basis. The units are kg emitted per kg fresh MC used.

[d] The term "recycle" refers to the information that, on a national basis, 75 percent of the MC contained in waste solvent streams is recovered by recycling companies and resold for further use in cleaning. This results in a reduction in the amount of fresh solvent required for a given cleaning application, but the percentage of fresh solvent usage that is ultimately emitted by the cleaning process is higher. See Appendix E for calculations and assumptions made for uncontrolled emission factor with recycle.

[e] The relative controlled solvent usage is defined as the amount of fresh solvent used by an uncontrolled cleaner to perform a given cleaning job.

EF = Emission Factor

TABLE 29. NATIONAL EMISSIONS OF MC FROM ORGANIC SOLVENT CLEANERS, (1987)[d]

Type of Cleaner	Emissions (Mg/yr)		
	Uncontrolled Cleaners[a]	Controlled Cleaners[b]	Total
Cold Cleaner	9,300	1,480	10,800
Carburetor Cleaner	0	1,620	1,620
Photoresist Stripper	6,540	1,110	7,650[c]
OTVC	3,230	1,490	4,720
In-Line Vapor Cleaner	1,370	430	1,800

Source: Reference 4.

[a] Refer to cleaners that are uncontrolled at baseline.

[b] Refers to cleaners assumed to be controlled with CTG Control System B at baseline.

[c] This includes baseline emissions at 755 Mg/yr reported by nine large photoresist stripping operations responding to EPA questionnaires in addition to emissions of 6,890 Mg/yr calculated using the approach described in this section.

[d] It is important to note that these emission estimates are based on 1983 solvent cleaning consumption estimates for fresh MC. Regulatory activity affecting the use of MC in recent years has resulted in fluctuations and decline in the use of MC. Methylene chloride 1991 consumption estimates for the solvent cleaning industry have been reduced 48 percent when compared with the consumption estimates that were used to calculate the emissions presented in Table 29.[1,3,4] However, it is not possible to apply this percent consumption decrease directly to scale down the 1983 national emission estimates to produce 1991 estimates, because other factors, such as county MC usage patterns and the distribution of controlled versus uncontrolled cleaners, have changed over time. These changes would need to be quantified to estimate emissions from current consumption.

uncontrolled solvent cleaners, and solvent usage for regulated counties was multiplied by an emission factor that represented controlled solvent cleaners.

REFERENCES FOR SOLVENT CLEANING AND PHOTORESIST STRIPPING

1. "Chemical Profile--Methylene Chloride," Chemical Marketing Reporter, 241(9):42, March 2, 1992.

2. "Solvent Use in Electronics--1989," Memorandum from Steve Risotto, Center for Emission Control, HSIA, Washington, DC, to Chris Sarsony, Radian Corporation, Research Triangle Park, NC, April 2, 1992.

3. Alternative Control Technology Document - Halogenated Solvent Cleaners, EPA-450/3-89-030, U.S. Environmental Protection Agency, Research Triangle Park, NC, August 1989.

4. "Calculation of Number of Organic Solvent Cleaners and Solvent Emissions and Use Per Model Plant," Memorandum from R.C. Mead and R.F. Pandullo, Radian Corporation, Research Triangle Park, NC, to D. Beck, U.S. Environmental Protection Agency, Research Triangle Park, NC, September 8, 1987.

5. "Documentation of Emissions and Long-Term Exposure Model Inputs for the Organic Solvent Cleaning Source Category," Memorandum from R.C. Mead and R.F. Pandullo, Radian Corporation, Research Triangle Park, NC, to D. Beck, U.S. Environmental Protection Agency, Research Triangle Park, NC, September 3, 1987.

AEROSOL PACKAGING AND DISPENSING

An aerosol is a suspension of solids or liquid particles in a gas that consists of a liquid and vapor phase. The liquid phase comprises active ingredients, solvent, and liquefied propellant or co-solvent. The vapor phase consists of the propellant that provides the positive pressure to expel the contents of the container.[1]

In an aerosol dispensing system, a liquid propellant keeps the pressure in the container constant as the product is being consumed. Methylene chloride is used in the aerosol industry as a solvent, co-solvent, and vapor pressure suppressor.[2] A solvent with the properties of MC acts to bring the active ingredient into solution with the propellant. A co-solvent is often used with MC when it is desirable to also have another liquid that is not miscible with the propellant (e.g., water). Methylene chloride, because of its high vapor pressure, high boiling point, formulation compatibility, and ability to depress the vapor pressure of high-pressure propellants, decreases the flammability of the formulation mixture and enhances dispersion of the aerosol spray.

Use of MC in aerosols has decreased because of Federal government labeling requirements on consumer goods containing potential carcinogens, such as MC. Substitutes for MC in aerosols with diversified uses include 1,1,1-trichloroethane (TCA), tetrachloroethane, mineral spirits, and water-soluble formulas.[3] Substitutes with limited uses include 1,1,2-trichloro-1,2,2,-trifluoroethane.[3]

Air emissions of MC result from packaging and consumer use of aerosols. The following sections discuss the aerosol packaging industry, aerosol packaging and dispensing processes, emission sources, and emission estimates of MC from aerosol packaging and use.

Aerosol Packaging Industry

There are an estimated 212 aerosol packaging companies, which consumed an estimated 8 percent (10,000 Mg) of the 1991 MC production.[4,5] The U.S. MC demand for use in aerosol products has decreased steadily in recent years

because of environmental and occupational health concerns.[5] In 1987, aerosol products containing MC reported by aerosol packaging companies included insecticides, cleaners, lubricants, spot removers, paints, primers, adhesives, sealants, enamels, and mold releases.[6] Since 1987, however, there has been a trend away from the use of MC toward the use of TCA in aerosol products. The U.S. production of TCA, however, is being phased out under the Montreal Protocol and the 1990 amendments to the Clean Air Act, as TCA is considered an ozone depleting substance. There has been a corresponding trend back toward the use of MC.[7] Solvent TCA production levels will be cut incrementally until phase-out at the end of the decade, and buyers are subject to an escalating excise tax to discourage use. Increasing pressure to find other solvent substitutes for MC is being exerted by the new OSHA-proposed MC exposure standard discussed in Section 3.[3]

Industry reports that the product types and range of MC content within the aerosol products reported in 1987 are similar to the MC content range within aerosol products today, but that the number of products, and volume of MC consumed by the aerosol industry has been dramatically reduced.[7] Consumption reduction of MC by the aerosol industry since 1987 is exhibited by Chemical Marketing Reports Chemical Profile of MC in 1991, which estimates a 49 percent reduction in aerosol consumption from 1988 (19,600 Mg/yr) to 1991 (10,000 Mg/yr).[5,8] A national list of 212 aerosol packagers that have the potential to consume chlorinated solvents was developed by the EPA in 1987. Appendix C includes the names, locations, and product types packaged at these facilities when data were available.[4]

Section 114 questionnaires containing questions about MC and other chlorinated solvent emission sources and methods of recovery or control were distributed to nine aerosol packaging companies. Eight of the nine companies' questionnaire responses (containing information on 11 facilities) were analyzed. Table 30 contains a list of aerosol product types reported in the Section 114 questionnaire responses and information on the amount of chlorinated solvent reported for each product type.[4]

TABLE 30. WEIGHT PERCENT OF METHYLENE CHLORIDE IN AEROSOL PRODUCT TANKS, REPORTED BY SECTION 114 QUESTIONNAIRE RESPONDENTS (1987)

	Methylene Chloride	
Product Type	Mean[a]	Range[b]
Spray Paints[c]	27	5-40
Insecticides	19	10-42
Lubricants[d]	17	5-55
Cleaners[e]	26	5-50
Adhesives	37	5-50
Paint Strippers	80	75-85

Source: Reference 4.

[a] The eight Section 114 questionnaire recipients were asked to provide the typical concentrations (weight percent) of chlorinated solvent in each product type. The value is the value of the reported typical concentrations.

[b] The range of typical concentrations reported by the eight questionnaire respondents.

[c] Includes enamels, coatings, primers, and rust inhibitors.

[d] Includes mold release agents and metal cutting fluids.

[e] Includes solvent degreasing cleaners, automotive, household, and electrical contact cleaners.

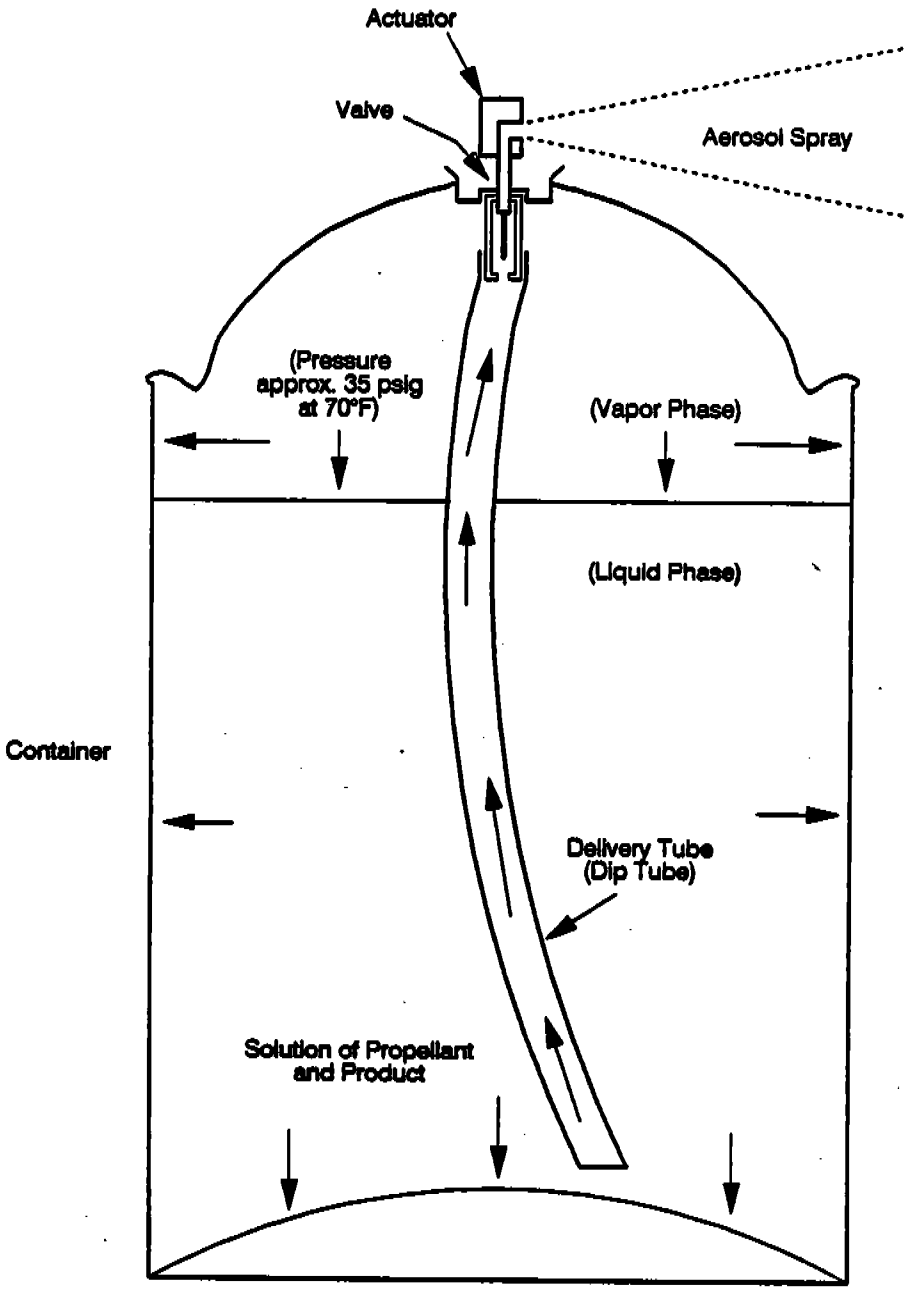
Process Description

Aerosol Packaging--

Aerosol packaging processes are batch-mode operations. Many companies contract out aerosol packaging because of high plant costs. Some companies fill other companies' products as well as their own, while others only fill aerosols for other companies.[3] Methylene chloride is generally supplied by an outside chemical supplier, who delivers the solvent either by a tank truck or by drums. Storage of MC can be in fixed-roof or pressurized tanks, or in the drums (normally 55-gallon drums) in which the solvent was delivered.[4]

The initial stage of the aerosol packaging process involves the mixing of the aerosol product ingredients in mixing tanks of varying sizes. Ingredients, including MC, are either pumped from storage tanks or are poured directly from storage drums. Ingredients are added according to volume or weight. In order to minimize solvent loss due to volatilization, some facilities add the solvent directly to the aerosol cans at the time of filling rather than adding the solvent at the mixing tank stage. Mixing tank ingredients, after being properly mixed, are either pumped or transported to the aerosol can filling lines.[4]

Aerosol can filling involves filling with product, inserting the valve stem and valve, adding propellant, and sealing the product in the can. Empty aerosol cans are conveyed to an automatic filler that uses filling nozzles to deliver the product into the empty cans. Filled aerosol cans are then conveyed to a point where the valve stems and valves are inserted into the can. The cans, still unsealed, are conveyed to a special "explosion-proof" room where the propellant is added to the cans. The cans are then sealed under pressure and conveyed to a point at which actuators are placed onto the stems. To complete the aerosol packaging process, each can is weighed to ensure weight requirements are met, submerged in a hot bath to check for leaks, and washed, labeled, capped, and packaged according to the particular requirements of the product.[4] An example of a typical aerosol spray system is illustrated in Figure 17.[9]



Source: Reference 9

4415338R

Figure 17. Typical aerosol spray system.

Aerosol Dispensing--

As illustrated in Figure 17, the aerosol spray dispensing product comprises of four basic components: propellant, product, container, and valve. A propellant is defined by the Department of Transportation as "a material which can expel the contents of an aerosol container at room temperature."[3] Propellants typically are liquefied gases with vapor pressures greater than atmospheric pressure, which enables product contents to be forced from the container when the valve is activated at room temperature. The product contains the solvent, co-solvent, and active ingredients that, in conjunction with the propellant, enable the product to perform its desired end function.

The dispensing process of an aerosol spray is achieved by depressing the actuator, which causes depression of the valve, releasing a solution of propellant and product to the air. The pressurized container allows propellants that are gaseous at atmospheric pressure to exist predominantly as a liquid. As the propellant is released, it converts to gas phase and disperses the product. Some gas-phase propellant remains in the container head space, while the rest is in equilibrium with the product. Examples of propellants used in aerosol dispensing systems include hydrocarbons, dimethyl ether (DME), hydrochlorofluorocarbon-152a (HCFC 152a), and compressed gases.[10]

Emission Sources

Aerosol Packaging--

Aerosol packaging process MC emissions may occur from the following general sources: storage tanks, handling operations (e.g., mixing tank loading), equipment leaks, wastewater, and accidental releases. Emissions from these sources to the atmosphere would be from the following:

- Building openings, which would release MC emissions along with other indoor air emissions;
- Process vents, which would release MC emissions directly from the source to the atmosphere, without dilution by other indoor air emissions; and

- Outdoor sources, which would include equipment used to store and transfer MC, and on-site wastewater treatment facilities.[4]

These MC emission sources from aerosol packaging processes are discussed in the following paragraphs.

Methylene chloride storage tank emissions occur from breathing and working losses. Breathing losses result from changes in barometric pressure and temperature, and working losses result from volumetric changes in the tank from filling or dispensing of stored solvent. Indoor storage tanks are expected to have minimal or negligible breathing losses because of indoor temperature controls that minimize diurnal temperature variation. Pressurized tanks that have pressures greater than the atmospheric pressure are not expected to have MC emissions.[4] An example calculation of storage tank emissions, using AP-42 methodology for storage of organic liquids, is presented in Appendix A. Accurate estimation of storage emissions, as discussed in Section 4, requires site-specific information.

Handling and transfer emissions result from filling and mixing tank operations, aerosol can filling, and aerosol can washing. Filling and mixing tank emissions are considered to be the most significant source of MC handling emissions, and occur as the solvent is added (i.e., the filling) to the mixing tank (as other ingredients are added), and during the mixing process itself.[4] Estimation of handling and transfer emissions require the summation of emissions from handling and transfer operations specific to a facility. The reader is referred to AP-42, Section 4.0, Evaporation Loss Sources, for use as a guideline in the estimation of these emissions.

Methylene chloride equipment leaks result from process equipment components leaking in a liquid or gaseous state. These losses may occur intermittently or continuously.[4] An example calculation for estimating emissions as a result of equipment leaks is presented in Appendix A. Methylene chloride emissions that occur during on-site treatment and disposal of wastewater, liquid waste, or solid waste are considered to be secondary emissions.[4] Appendix A contains a simplified example calculation for estimating MC emissions from secondary wastewater treatment processes. For a more detailed and accurate methodology for estimating MC emissions from

secondary wastewater treatment, the reader is referred to the EPA document "Industrial Wastewater Volatile Organic Compound Emissions--Background Information for BACT/LAER Determinations. EPA-450/3-90-004." As with MC production, methodologies to estimate emissions require site-specific parameters in order to represent emission potential accurately.

Methylene chloride emission controls that may be incorporated in the aerosol packaging process include storage tank refrigerated condensers, process vent carbon adsorbers, and process vent refrigerated condensers. The use of refrigerated condensers lowers the vapor pressure, and therefore, the emission potential of the solvent. Additional control techniques with associated control efficiencies for emissions of chlorinated solvents from aerosol packagers is shown in Table 31.[4] Methylene chloride evaporation losses may also be controlled by the use of external or internal floating-roof tanks in place of fixed-roof tanks. Control efficiencies for these tanks vary according to the size of the tank and the type of seal employed.

Aerosol Dispensing Processes--

Emission of MC from aerosols result from the use of the product, and the crushing, compacting, leakage, corrosion, and permeation of the aerosol container that contains MC. Methylene chloride emissions from the consumption of aerosol products result from the volatilization of suspended droplets or by evaporation from sprayed surfaces.

Methylene chloride emission controls that may be employed include minimization of MC content and integrity maintenance of the aerosol container.

Emission Estimates

Aerosol Packaging Process Emission Estimates--

Methylene chloride emission estimates from aerosol packaging processes, based on Section 114 responses in 1987, were determined and documented in 1988 under a previous EPA project.[12] Emissions were estimated for storage tanks, handling operations, indoor and outdoor equipment leaks, secondary sources, and accidental releases. The calculation procedures are documented in another

TABLE 31. CONTROL TECHNIQUES FOR EMISSIONS OF CHLORINATED SOLVENTS FROM AEROSOL PACKAGERS (1988)

Emission Source	Control Technique	Emission Reduction Efficiency (%)
Storage tank	Refrigerated condenser	95
Mixing Tanks (Handling)	Carbon Adsorption[a]	95
	Refrigerated condenser[a]	95
Equipment Leaks		
Pump Seals (packaged and mechanical)	Monthly LDAR	61
Flanges	None Analyzed	--
Valves (liquid)	Monthly LDAR	59
Valves (gas)	Monthly LDAR	78
Sample Connections	Closed-purge sampling	100
Open-Ended Lines	Caps on open ends	100
Secondary Sources	None analyzed	0

Source: Reference 4.

[a] Control option also includes covering the mixing tank and installing ductwork from the mixing tank to the adsorber or condenser to recover chlorinated solvent emissions.

LDAR = Leak Detection and Repair

memo.[12] Estimates of emissions from 10 of the 11 1987 Section 114 responses were used to determine MC emissions for the remaining 184 facilities that used MC in their aerosol products filled.[12] Table 32 presents the emission estimates from these 10 Section 114 responses.

The MC consumption reported in the 10 Section 114 responses was 8,400 Mg/yr, and reported emission estimates were 81.4 Mg/yr. An emission factor for the entire aerosol packaging process, based on the uncontrolled aggregate emissions/consumption for these 10 questionnaire respondents, is 0.01 Mg/Mg MC consumed (19.4 lb/ton consumed).[13]

Aerosol Dispensing Process Emission Estimates--

There are no MC emission estimates available for the use of aerosol products at this time. Trends away from MC use in aerosol products, however, should reduce emissions proportionately to the reduction of use. Methylene chloride content is ultimately considered to be released to the environment via differing media (e.g. soil, air, and water). The only emission factor found in the literature for aerosol products use was the worst-case assumption that MC emissions are 1 kg/kg MC contained in product applied (2000 lb/ton MC contained in product applied).[13]

TABLE 32. ESTIMATED METHYLENE CHLORIDE EMISSIONS FROM AEROSOL PACKAGERS IN 1987

Company Name	Location	Methylene Chloride Emissions (Mg/yr)					
					Equipment Leaks		
		Total	Storage	Handling	Outdoor	Indoor	Secondary
New York Bronze Powder Co.	Elizabeth, NJ	18.1	6.6	8.3	NR	3.2	NR
New York Bronze Powder Co.	Taylor, PA	24.9	2.6	18.1	NR	4.2	NR
Percy Harms corporation	Wheeling, IL	0.6	0[a]	0.4[b]	NR	0.2	NR
Plaze, Inc.	St. Louis, MO	2.4	0.5	0.6	NR	1.3	NR
Seymour of Sycamore	Sycamore, IL	5.8	0.2	5.5[c]	0.1	<0.1	NR
Sherwin-Williams Co.	Anaheim, CA	4.5	0.3	4.1	NR	0.1	NR
Sherwin-Williams Co.	Bedford Heights, OH	9.6	2.0	6.1	0.5	1.0	NR
Sherwin Williams Co.	Elk Grove village, IL	0.8	0.2	0[d]	0.3	0.3	NR
Speer Products	Memphis, TN	13.2	1.4	11.8	0.03	0.2	NR
Zep Manufacturing	Atlanta, GA	1.5	0.2	0.8	0.3	0.2	<.01
Total		81.4	14.0	55.5	1.2	10.7	<.01

Source: Reference 4.

[a] A pressurized tank is used to store MC at this facility.

[b] Estimate includes 0.2 Mg/yr for aerosol can washing.

[c] Estimate includes 1.3 Mg/yr for aerosol can washing.

[d] No mixing operations are used at this facility because aerosol ingredients are added directly from storage to the product containers.

NR = Not reported.

REFERENCES FOR AEROSOL PACKAGING AND DISPENSING

1. J.M. Reynolds, Cost Effective Replacements for Chlorinated Solvents and Glycol Ethers, Spray Technology and Marketing, Industry Publications, Inc., Fairfield, NJ, July 1991.

2. L.W. Patrick, Spray Technology and Marketing. Industry Publications, Inc., Fairfield, NJ, December 1991.

3. Occupational Safety and Health Administration Proposal for New Methylene Chloride Standard, 56 FR 57046-57047, 57068-57059, November 7, 1991.

4. "Emission Estimates and Controls from those Aerosol Packaging Facilities Responding to Section 114 Questionnaires," Memorandum from J. Martinez, R. Wassel, and G. Bockol, Radian Corporation, Research Triangle Park, NC, to the Methylene Chloride Source Category Investigations File, U.S. Environmental Protection Agency, Research Triangle Park, NC, October 13, 1987.

5. "Chemical Profile--Methylene Chloride," Chemical Marketing Reporter, 241(9):42, March 2, 1992.

6. "Documentation of Telephone Communications Concerning the Packaging and Localized Large-Scale Users of Aerosol Products," Memorandum from G. Bockol, Radian Corporation, Research Triangle Park, NC, to the Methylene Chloride Source Category Investigations File, U.S. Environmental Protection Agency, Research Triangle Park, NC, February 6, 1987.

7. "Methylene Chloride Use in Aerosols as Compared with 1987 Uses," Telephone communication from Joanne O'Loughlin, Radian Corporation, Research Triangle Park, NC, with Stephanie Kubik, Sprayon Products - Division of the Sherwin-Williams Company, March 25, 1992.

8. "Chemical Profile--Methylene Chloride," Chemical Marketing Reporter, February 20, 1989.

9. Volatile Organic Compound Emissions from Consumer and Commercial Products, Aerosol Products and Packaging Systems, Report to Congress, U.S. Environmental Protection Agency, Research Triangle Park, NC, December 1991.

10. Aerosol Products Workshop Summary, EPA Contract Number 68-02-3816, U.S. Environmental Protection Agency, Research Triangle Park, NC, and the Chemical Specialties Manufacturers Association, Washington, DC, October 24, 1990.

11. Industrial Wastewater Volatile Organic Compound Emissions -- Background Information for BACT/LAER Determinations, Chapter 4, EPA-450/3-90-004, U.S. Environmental Protection Agency, Research Triangle Park, NC, March 1990.

12. "Documentation of Baseline and Controlled Emission Parameters for Aerosol Packagers," Memorandum from E. Moretti, Radian Corporation, Research Triangle Park, NC, to the Methylene Chloride Source Category Investigations File, U.S. Environmental Protection Agency, Research Triangle Park, NC, January 19, 1988.

REFERENCES FOR AEROSOL PACKAGING AND DISPENSING (Continued)

13. Crosswalk/Air Toxic Emission Factor Data Base Management System (XATEF), Version 1.2, Diskette, U.S. Environmental Protection Agency, Research Triangle Park, NC, 1991.

MISCELLANEOUS

Approximately 5 percent of the total U.S. consumption of MC is for miscellaneous uses such as pesticide manufacture, photographic film processing, food processing, rubber cement and rubber accelerator manufacture, dye carrying, solid waste treatment, storage and disposal (TSD), and site remediation. Information gathered on pesticide manufacture and photographic film processing, the largest two MC users in this miscellaneous group, are discussed briefly in the following text. In addition, MC use in paints and coatings is outlined.

Pesticide Manufacturing

Methylene chloride may be used in two areas of the pesticide industry: (1) the manufacture of the pesticide; and (2) the formulation of the pesticidal materials with the necessary additives and inert carriers. Previous studies indicate that there are about 140 individual pesticide manufacturing facilities and 200 to 300 formulation plants located throughout the United States.[1,2] However, only a fraction of these may use MC.[1] One source estimates that 60 manufacturers/formulators use 10 million pounds of MC per year.[3] The location of these facilities or the amount, if any, of MC used by each were not verified in this study.

Methylene chloride has several applications in the pesticide manufacturing industry, including extraction, phase separation, purification, crystallization, and as a general transport solvent.[1] In the formulation of pesticide products, MC is sometimes used as a solvent to produce liquid products from granular active ingredients.[3] However, according to OSHA, there is an indication that MC usage in pesticide processes will soon be phased out because of health concerns. Products that have been substituted for MC included petroleum distillates, aqueous formulas, mineral spirits, and Agatane.[3]

Because of concern that some inert ingredients in pesticide products might cause adverse effects in humans or the environment, EPA developed a regulatory policy for inert ingredients.[4] The EPA divided the approximately 1,200 inert ingredients contained in pesticide products into four toxicity

categories: Lists 1 and 2 contain inerts of toxicological or potentially toxicological concern, and Lists 3 and 4 contain inerts of unknown toxicity or minimal concern. Methylene chloride is found on List 1. According to one pesticide formulator, List 1 chemicals can be used in pesticides as long as there is a warning on the product label, but to his knowledge, most of the constituents (such as chlorinated solvents) are no longer used in these products.[5]

Process Description--

The methods and exact technology for manufacturing pesticides varies considerably depending on the type of pesticide, but MC is frequently used in certain manufacturing steps. The pesticide industry employs the same unit processes and operations used in the chemical processing industry.[1] These include chemical reactions, filtering, separation operations, condensation, and drying. Production processes are usually carried out at ambient or slightly above ambient temperatures. Elemental chlorine is the raw material common to most pesticide production and is also frequently used to prepare other raw materials used for pesticide production.

Emissions--

Air emissions from pesticide facilities include particulates, gases, and vapors that may emanate from process equipment at each step of the manufacturing process.[1] Sources include process vents, storage, transfer, handling, equipment leaks, and wastewater. An inspection of one pesticide manufacturing facility revealed 26 potential sources (e.g., pellet mill, grinding unit, Munson mixer, bagging unit) of air pollution and 13 emission points (e.g., baghouse/cyclone, exhaust hoods, vents, and fans).[6] The MC emissions from three of these emission points can be found in Table 33.[6]

A comprehensive survey of typical control techniques and control levels for pesticide manufacturing/formulating was not undertaken. However, control techniques employed by several pesticide facilities using MC include condensers, recycling systems, and incinerators.[1] In general, applicable control techniques would be similar to those discussed in Section 4 for MC production.

TABLE 33. EMISSIONS FROM THREE POINTS AT A PESTICIDE FACILITY, 1988

Source	MC Emissions kg/yr (lb/yr)
Pellet Mill	334.7845 (738)
Marion Mixer	431.1803 (951)
Liquid Blending Area #1	12.4344 (27)

Source: Reference 6.

NOTE: Emission data are for one facility only and do not represent average emissions for all such sources, or total emissions for all sources.

Photographic Film Manufacture

Methylene chloride is a key component in the manufacture of cellulose triacetate-based films. It is used with heat to dissolve cellulose triacetate pellets in order to produce a fine, transparent layer which is rolled onto photographic paper. This layer must be transparent, durable, and flexible. Other materials used in the manufacturing process are plasticizers and small amounts of other solvents.[9] Virtually all still camera, graphic arts, and photographic films in use in the United States are cellulose triacetate-based. Substitutes for MC in this process have not been identified.[7]

Photographic film is currently produced at two sites. The largest facility is owned and operated by Eastman Kodak, with a facility located in Rochester, New York. This plant consumes approximately 4 Mg (9 million pounds) of MC annually.[8] Anitec Image Technology Corporation in Binghampton, New York uses approximately 0.9 Mg (1.9 million pounds).[9]

Process Description--

Several steps are required in the development of cellulose triacetate film. These steps are represented in Figure 18. Methylene chloride, plasticizers, cosolvents, and solid cellulose triacetate pellets are heated to produce a thin slurry called "crude dope." The dope is 60-65 percent MC by weight.[9] Vapors from this melting process are recovered by use of distillation equipment. The crude dope is then transported through the filtration process by use of a continuous screen, continuous wash, and multiscreen filters. This process is completed in a closed system; however, the continuous wash and multiscreen filters are changed four to five times per day.

The filtered dope is no longer considered crude at this point. It is fed into a receiving hopper that extrudes dope onto a large, polished cylindrical wheel, called a rollcaster. The dope is trimmed from the wheel, leaving a thin sheet or "web" that is dried at 121-138°C in an enclosed chamber. The MC is nearly completely evaporated in this process. The roll casting process is operated in a semi-enclosed manner, with some of the evaporating MC removed, and the remainder exhausted to the atmosphere.[9]

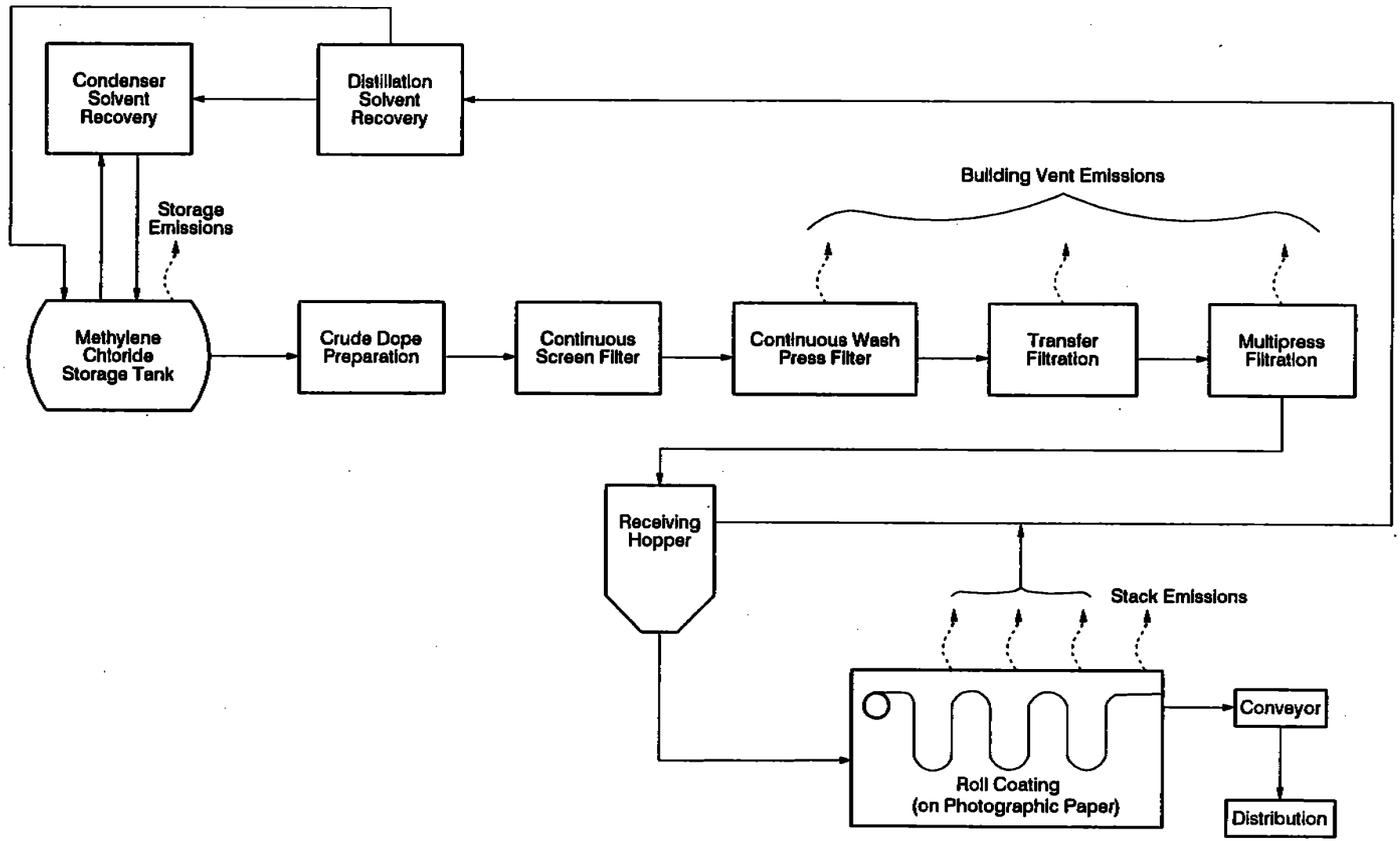


Figure 18. Photographic film process with MC emission sources.

Emissions--

Emissions from the manufacture of cellulose triacetate-based film can result from three basic stages of production: dope preparation, roll coating, and distillation or recycling. Numerous emission sources have been identified, but data have only been released for the most significant sources.

Dope production is the process of dissolving cellulose triacetate pellets. This is generally initiated by use of continuous mixers; however, batch mixers may be occasionally used. Emissions can be significant when these pellets are introduced to the mixing chamber during batch mixing. Current emission estimates for this activity are not available because the frequency of batch mixing has decreased markedly in recent years. Two other significant emission points in the dope production phase occur when filters for the continuous wash, transfer, and multipress filtration units are changed.[9,13] This process occurs approximately twice per day for each unit. The filters are removed from a cylindrical housing and allowed to evaporate. Emissions from this process have been estimated to be higher than all other dope production sources.

All other sources of MC emissions from dope production are associated with storage tanks and general building ventilation. The largest of these points are from "floor sweeps." These are ducted vents located near the filter housings that exhaust MC that accumulates near the floor.

Emissions from the roll coating machine represent over 90 percent of the emissions at a typical facility. During solvent evaporation of the film base web, local exhaust ventilation transports vapors to distillation and condenser recovery systems. This system recovers approximately 95 percent of the MC vapors in the process. The remaining 5 percent are released into the building, which has a ventilation design to rapidly move vapors from the floor and out of the building through vents or stacks. Large volumes of MC and the high temperatures at which the system operates account for the high level of emissions relative to other process functions. The building ventilation system, which serves primarily as a method to reduce workers' exposures, may also contribute to the building emissions.[9]

The other major process function in the manufacture of cellulose triacetate film is the distillation of recovered vapors. The major point of emissions from this source are likely to be storage tanks associated with the process.

Control technologies at the Kodak Rochester facility have included carbon adsorbers, scrubbers, condensers and vapor return. The exact locations of many of these controls have not been released. However, several control methods are used to recover MC. Carbon adsorbers have been applied at the roll coating machines exhaust air and building openings. Scrubbers have been applied in the solvent recovery systems. Reliable estimates of control efficiencies for these systems are not available. Other controls for fugitive emissions have included the selection of new valves with tighter seals and an increase in inspection and maintenance of existing potential process leaks. The pending OSHA regulations may significantly affect the type of controls that may be implemented.

Paints and Coatings

The paint and coatings industry encompasses a wide variety of products with many different end uses. A paint or coating is defined as a "liquid, liquefiable, or mastic composition that is converted to a solid, protective, decorative or functional adherent film by the application of a thin layer."[10]

Paints and coatings are produced by an estimated 390 facilities that consume 12,700 Mg of MC annually.[11] Paint and surface coatings are formulated by mixing three elements: synthetic polymer resins, which act as a binding agent, a dispersion medium (water or a volatile solvent), and pigments. Methylene chloride may be used in some products as a cosolvent in the dispersion medium to promote faster drying and dissolve binders during application. The main ingredients used in solvent-based paints are mineral spirits or petroleum naphtha products. Because MC reacts with dried paints on previously painted surfaces, its composition in paints is relatively low -- one or two percent by weight.[11]

For paints and coating products, recent emphasis on production of low-VOC products may increase the demand for additional MC use as a cosolvent

because MC may be exempted from regulation as a VOC under state regulations implementing the national ambient air quality standard for ozone.[12] The impact of these regulations, however, is highly speculative at this time.

Methylene chloride emissions data for paints and coatings formulation and use was not found.

REFERENCES FOR MISCELLANEOUS SECTION

1. Survey of Methylene Chloride Emission Sources, EPA-450/3-85-015, U.S. Environmental Protection Agency, Research Triangle Park, NC, June 1985.

2. T. Ferguson, Pollution Control Technology for Pesticide Formulators and Packagers, EPA-660/2-74-094. U.S. Environmental Protection Agency, Research Triangle Park, NC, January 1975.

3. Occupational Safety and Health Administration Proposal for New Methylene Chloride Standard, 56 FR 57053, 57069, November 7, 1991.

4. Inert Ingredients in Pesticide Products; Policy Statement, 52 FR 13305, April 22, 1987.

5. Written communication from Jim Case, S.C. Johnson & Son, Inc., Racine, WI, to Candace Blackley, Radian Corporation, Research Triangle Park, NC, September 10, 1991.

6. High Risk Point Source Study for the Formulation of Pesticides and Agricultural Chemicals at Southern Mill Creek Products, Inc. (SMCP), Tampa, Florida, Air Toxics Section of the Environmental Protection Commission of Hillsborough County, pp. 1-9, Appendix E, May 1990.

7. S.A. Walata and R.M. Rehm, Source Characterization and Control Technology Assessment of Methylene Chloride Emissions from Eastman Kodak Company, Rochester, New York, EPA-600/2-89-043. U.S. Environmental Protection Agency, Research Triangle Park, NC, 1989.

8. White Paper -- Methylene Chloride. Halogenated Solvents Industry Alliance, Washington, DC, February 1989.

9. Occupational Safety and Health Administration Proposal for New Methylene Chloride Standard, 56 FR 57066, November 7, 1991.

10. "Paint--Tests for Formulated Products and Applied Coatings," 1990 Annual Book of ASTM Standards, Volume 06.01, R. Storer, et al. eds., American Society for Testing and Materials, Philadelphia, PA, 1990.

11. Occupational Safety and Health Administration Proposal for New Methylene Chloride Standard, 56 FR 57102, November 7, 1991.

12. "Summary of Information Obtained Through Telephone Contact with Formulators, Distributors, and Users of Methylene Chloride-Based Paint Strippers," Memorandum from I. McKenzie, Radian Corporation, Research Triangle Park, NC, to the Paint Stripping Facilities Project File, U.S. Environmental Protection Agency, Research Triangle Park, NC, January 22, 1987.

SECTION 6
SOURCE TEST PROCEDURES

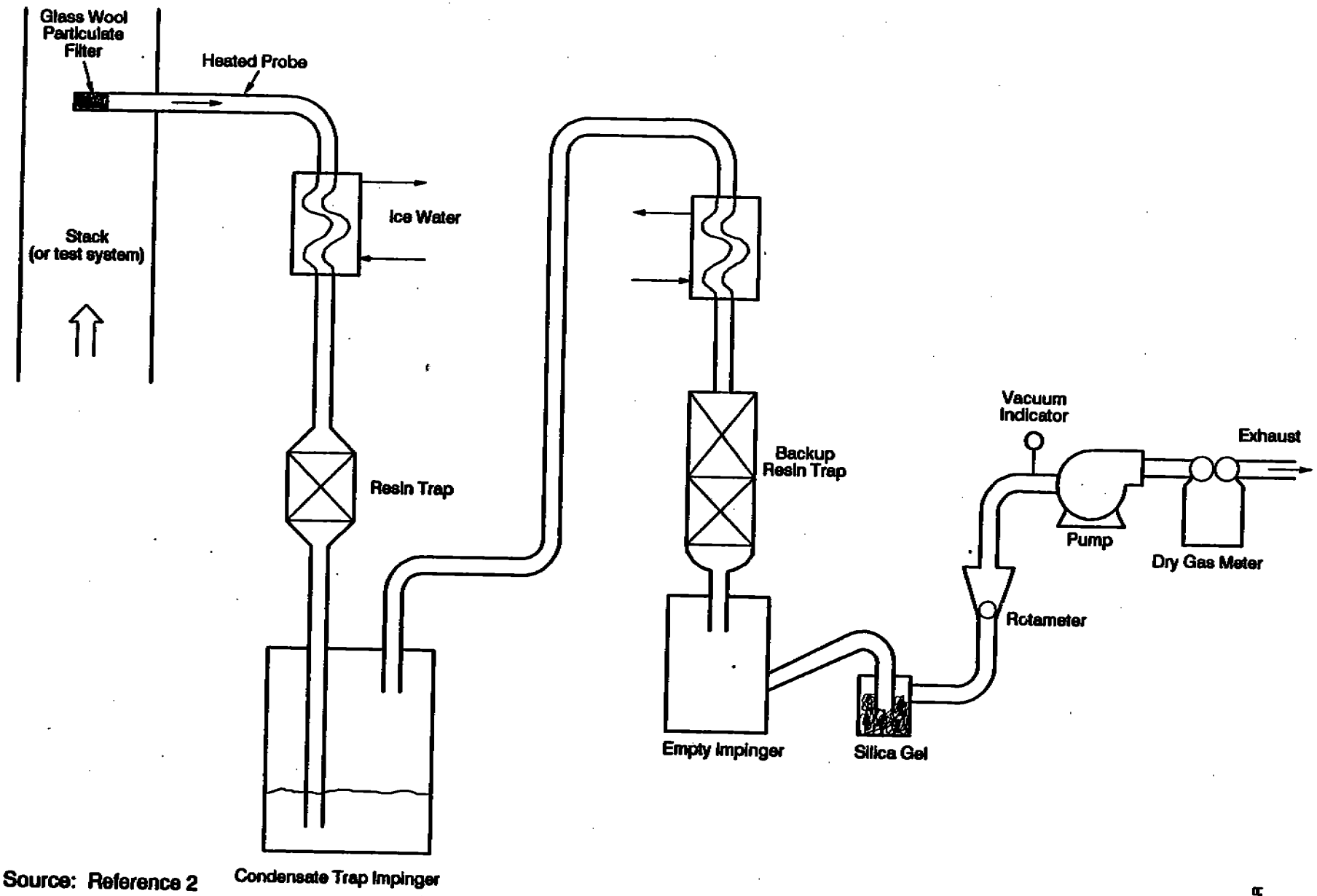
Methylene chloride emissions can be measured by a combination of the following methods:[1] (1) EPA Method 0030; (2) EPA Method 5040; (3) EPA Method 5041; (4) EPA Method 18; (5) Compendium Method TO-1; (6) Compendium Method TO-2; (7) Compendium Method TO-14; and (8) NIOSH Method 1005. Each of these methods is discussed in the following paragraphs.

EPA METHOD 0030

EPA Method 0030, which was published in EPA Report No. SW-846[2] in November, 1986, is a sampling method used to determine the destruction and removal efficiency (DRE) of volatile principal organic hazardous constituents (POHCs) from stack gas effluents from stationary sources. EPA Method 0030 is used for sampling MC emissions from stack gas effluents.[1] This methodology is applicable for sampling volatile constituents with boiling points between 35°C and 100°C; the boiling point of MC is within this range.

In Method 0030, a sample of effluent gas is withdrawn from an emission source using a glass-lined probe and a volatile organic sampling train (VOST). The gas stream is cooled through a water-cooled condenser and volatile POHCs are collected on a pair of sorbent resin traps, the first containing Tenax® and the second containing Tenax® and petroleum-based charcoal. Liquid condensate is collected in an impinger placed between the two resin traps. A schematic diagram of the VOST system is shown in Figure 19.

The sensitivity of this method depends on the level of interferences in the sample and the presence of detectable levels of volatile POHCs (in this case, MC) in the blanks. Interferences can arise from contamination of sorbent traps prior to or after use in sample collection.[3] Exposure of the sorbent materials to solvent vapors prior to assembly can be one source of interferences. Because MC is a common field recovery solvent, laboratory solvent, and laboratory air contaminant, contamination of the sorbent traps from these sources can also be a problem. Exposure to significant concentrations of volatile POHCs such as MC in the ambient air at chemical plants and other sources previously discussed is another potential



Source: Reference 2

Figure 19. Schematic of volatile organic sampling train (VOST).

6227123R

interference. A sufficiently high background level in the source can make it impossible to determine trace quantities in the samples.

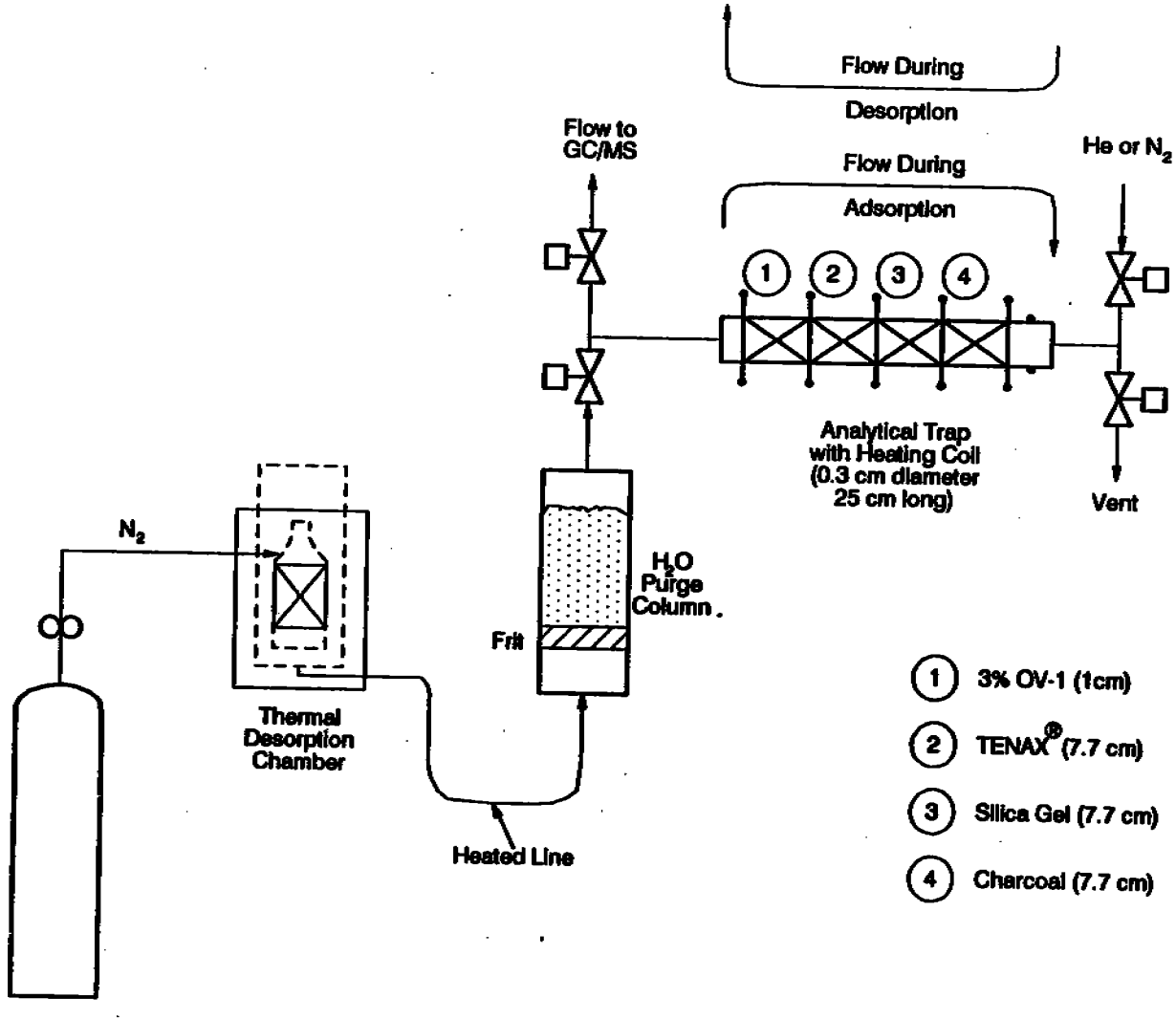
EPA METHODS 5040 AND 5041

EPA Methods 5040 and 5041 are both used to analyze POHCs collected from stack gas effluents of stationary sources. Method 5040 was published in the November 1986 Report No. SW-846.[2] Methods 5040 and 5041 are the primary analytical methods for determining the concentration of MC in stack emissions sampled using the VOST Method, EPA Method 0030.[1] The combination of Methods 0030/5040 or 5041 is intended for trace levels of organic compounds in emissions (i.e., ppb, low ppm). If the concentration in the source is hundreds of ppm or percentage levels, these methods are not appropriate. Because the majority of gas streams sampled using VOST will contain a high concentration of water, the analytical method is based on the quantitative thermal desorption of volatile POHCs from the Tenax® and Tenax®/charcoal traps. Purge-and-trap gas chromatography/mass spectrometry (GC/MS) is used to perform the analysis. The major difference between Methods 5040 and 5041 is the analytical column required. In Method 5040, a packed glass column is used; a Megabore® capillary column is used in Method 5041.

A schematic diagram of the analytical system is shown in Figure 20. The sorbent cartridges are spiked with internal standards and surrogates, thermally desorbed, and the VOCs are trapped on an analytical adsorbent trap directed into the GC/MS. The volatile POHCs are separated by temperature-programmed GC and detected by low-resolution MS.

The concentrations of volatile POHCs are calculated using the internal standard technique. Sample trains obtained from the VOST should be analyzed within two to six weeks of sample collection. The desired target detection limit of these methods is 0.1 nanogram per liter (ng/ℓ) (20 ng on a single pair of traps).

As with VOST sampling, solvent contamination can occur with analysis. Therefore, appropriate use of laboratory and field blanks is crucial for obtaining accurate quantitative values. Other concerns that may need to be addressed when using Methods 5040 and 5041 include saturation of the sampling



Source: Reference 3

3103963R

Figure 20. Schematic diagram of trap desorption/analysis system.

cartridges with the analyte (MC) or other compounds found in the stack gas emissions, and background interferences during analysis.

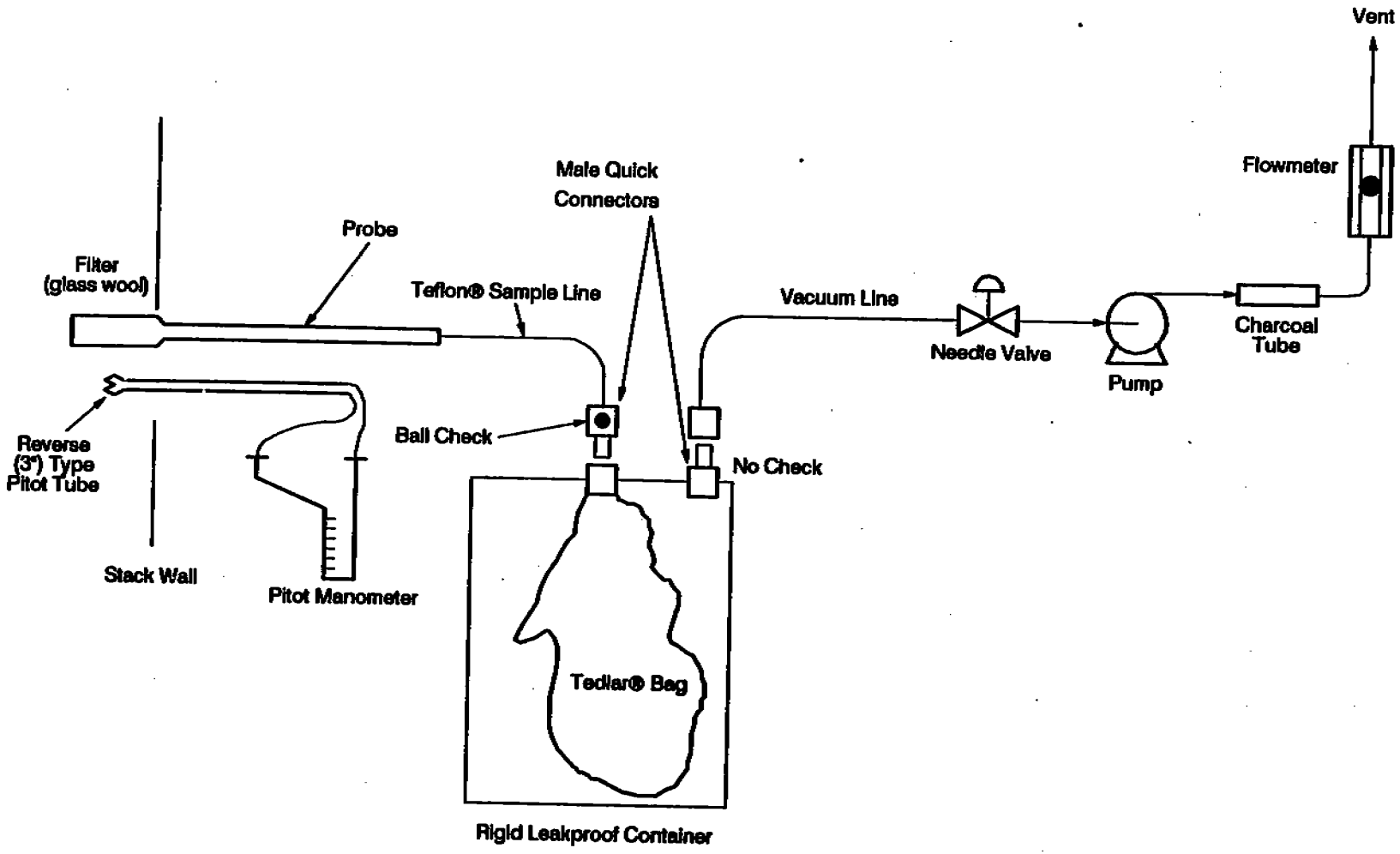
Interferences occur when something impairs the analyst's ability to make an accurate determination. In the case of a chromatographic method, this impairment is usually due to the presence of coeluting peaks that may arise from contamination of the sampling medium (sorbents). However, major sources of interference are constituents of the background during sampling. Many sources have a moderate to high background of hydrocarbons, some of which may coelute with MC and which may have common mass spectral ions. Other constituents of the source background may also provide chromatographic and mass spectral interference that will impair the ability of the analyst to perform an accurate determination.[4]

EPA METHOD 18

EPA Method 18 was announced in the Federal Register on October 18, 1983, and is published in Appendix A of 40 CFR Part 60.[5] It applies to the sampling and analysis of approximately 90 percent of the total gaseous organics emitted from an industrial source.[1] Method 18 is an alternative sampling and analytical method for VOCs (including MC). It can be used for single analytes or for a small number of multiple analytes. Method 0030, combined with Methods 5040 or 5041, has a broader application to a wider range of analytes and may, therefore, be preferred if there is a need to analyze for many organics. Method 18 has been used extensively for testing emissions from the chemical industry and other source categories emitting VOC.

In Method 18, a sample of the exhaust gas to be analyzed is drawn into a Tedlar® or aluminized Mylar® bag, as shown in Figure 21.[5] The bag is placed inside a rigid, leakproof container and evacuated. The bag is then connected by a Teflon® sampling line to a sampling probe (stainless steel, Pyrex® glass,or Teflon®) at the center of the stack. The sample is drawn into the bag by pumping air out of the rigid container.

The sample is then analyzed by GC coupled with flame ionization detection (FID). Based on field and laboratory studies, the recommended time limit for analysis is within 30 days of sample collection.[3] The GC operator



Source: Reference 5

Figure 21. Intergrated bag sampling train.

1181726R

should select the column and GC conditions that provide good resolution and minimum analysis time for MC. Zero grade helium or nitrogen should be used as the carrier gas at a flow rate that optimizes the chromatographic resolution.

The peak areas corresponding to the retention times of MC are measured and compared to peak areas for a set of standard gas mixtures to determine the MC concentrations. The detection limit of this method ranges from about 1 part per million (ppm) to an upper limit governed by the FID saturation or column overloading. However, the upper limit can be extended by diluting the stack gases with the inert gas or by using smaller gas sampling loops.

When access to the sampling location is difficult, an alternative sampling method described in Section 7.4 of EPA Method 18 may be preferred.[3]

COMPENDIUM METHODS TO-1, TO-2, AND TO-14

Compendium Methods TO-1, TO-2, and TO-14 are sampling and analytical methods used to determine VOCs such as MC in ambient air, and can be found in the "Compendium of Methods for the Determination of Toxic Organic Compounds in Ambient Air."[6] These methods are not appropriate for source monitoring.[1]

Method TO-1

Method TO-1 is used to collect and determine volatile, non-polar organics that can be captured on Tenax® and determined by thermal desorption techniques. In this method, air is drawn through a cartridge containing 1-2 g of Tenax®. The cartridge is analyzed in the laboratory for MC and purged with an inert gas into a GC/MS system. Only capillary GC techniques should be used. The GC temperature is increased through a temperature program and the compounds are eluted from the column on the basis of boiling points. The MS identifies and quantifies the compounds by mass fragmentation patterns. Compound identification is normally accomplished using a library search routine on the basis of GC retention time and mass spectral characteristics.

Method TO-2

In Method TO-2, air is drawn through a cartridge containing 0.4 g of a carbon molecular sieve (CMS) adsorbent. The cartridge is analyzed in the laboratory by flushing with dry air to remove adsorbed moisture and purging the sample with helium while heating the cartridge to 350-400°C. The desorbed organics (such as MC) are collected in a cryogenic trap and flash-evaporated into a GC/MS system. Only capillary GC techniques should be used. The GC temperature is increased through a temperature program and the compounds are eluted from the column on the basis of boiling points. The MS identifies and quantifies the compounds by mass fragmentation patterns. Compound identification of MC is normally accomplished using a library search routine on the basis of GC retention time and mass spectral characteristics.

Method TO-14

Method TO-14 is based on the collection of whole air samples in SUMMA® passivated stainless steel canisters for analysis of VOCs (MC) in ambient air. A sample of ambient air is drawn through a sampling train of components that regulate the rate and duration of sampling into a pre-evacuated SUMMA® passivated canister. The canister is attached to the analytical system. Water vapor may be reduced in the gas stream by a Nafion® dryer and VOCs are concentrated by collection into a cryogenically-cooled trap. The cryogen is removed and the temperature of the sample raised to volatilize the sample into a high-resolution GC. The GC temperature is increased through a temperature program and the compounds are eluted from the column on the basis of boiling points into a detector.

The choice of detector depends on the specificity and sensitivity required by the analysis. Gas chromatography ultimately relies on retention time for identification of compounds. In many cases, this use of retention time is enhanced by the information from a selective gas chromatographic detector. However, it cannot be determined from GC alone whether coelution of compounds is occurring, or whether a particular peak represents a particular compound. When mass spectrometry is used as the detector, a mass spectrum can provide compound-specific information and can show whether other compounds are present at a given retention time.[4]

In Method TO-14, a capillary column with methyl silicone coating, or equivalent is specified for detecting MC.[6] A wider Megabore® column can be used as long as the system meets user needs. Compounds have been successfully measured at the parts per billion by volume (ppbv) level using this method.

NIOSH METHOD 1005

The NIOSH methods are used to measure ambient air in workplace environments. NIOSH Method 1005, which was published in the 1985 "NIOSH Manual of Analytical Methods", is used to measure MC in the workplace.[7]

In NIOSH Method 1005, air samples are collected with solid sorbent tubes containing coconut shell charcoal. A personal sampling pump is used to collect 1.5 - 2.5-ℓ air samples at a flow rate of 0.01 to 0.2 ℓ/min. Samples are desorbed with carbon disulfide and analyzed by GC equipped with an FID. The column specified in NIOSH Method 1005 is a 3.0 m x 3 mm stainless steel, 10% SP-1000 on 80/100 mesh Chromosorb® W-HP, or equivalent.[7] The amount of MC in a sample is obtained from the calibration curve in units of milligrams per sample. The working range of NIOSH Method 1005 is 100 to 3000 ppm for a 1-ℓ air sample. The method is applicable to ceiling determinations.

REFERENCES FOR SECTION 6

1. Screening Methods for the Development of Air Toxics Emission Factors, EPA-450/4-91-021, U.S. Environmental Protection Agency, Research Triangle Park, NC, September 1991.

2. Test Methods for Evaluating Solid Waste, 3rd Ed., Vol. 1B: Laboratory Manual, Physical/Chemical Methods, EPA Report No. SW-846, U.S. Environmental Protection Agency, Washington, DC, November 1986.

3. Locating and Estimating Air Emissions from Sources of Styrene, Interim Report, EPA-450/4-91-029, U.S. Environmental Protection Agency, Research Triangle Park, NC, September 1991.

4. Telephone communication between Candace Blackley and Joan Bursey, Radian Corporation, Research Triangle Park, NC, March 26, 1991.

5. Code of Federal Regulations (Title 40, Part 60, Appendix A): "Method 18: Measurement of Gaseous Organic Compound Emissions by Gas Chromatography," 40 CFR 60, Appendix A.

6. Compendium of Methods for the Determination of Toxic Organic Compounds in Ambient Air, U.S. Environmental Protection Agency, Research Triangle Park, NC, May 1988.

7. NIOSH Manual of Analytical Methods, 3rd Ed., Volume 2, U.S. Department of Health, Education, and Welfare, National Institute for Occupational Safety and Health, Cincinnati, Ohio, 1987.

APPENDIX A

EXAMPLE PROCEDURES TO ESTIMATE METHYLENE CHLORIDE EMISSIONS

The purpose of this appendix is to present general example equations or calculation procedures and assumptions that may be used to estimate methylene chloride (MC) air emissions from differing sources. This appendix describes estimation procedures for three types of emission sources: (1) storage; (2) equipment leaks; and (3) secondary sources (wastewater).

The following sections describe example estimation procedures for emission sources.

SECTION 1 - STORAGE EXAMPLE

The following storage loss equations are presented for estimating fixed-roof breathing loss and fixed-roof working losses for a vertical tank storing methylene chloride. The following example equations are from the EPA Publication No. AP-42 emission factors for storage of organic liquids, Supplement E.[1] The purpose of their presentation is to alert the reader to the site-specific and chemical property considerations that are required for the estimation of MC emissions from storage. The equations presented are based on the following general points:

- Equations are for estimating standing storage and working losses for a vertical, fixed-roof storage tank.[1]
- Storage tank emissions from loading and unloading MC are accounted for in the working loss equation.

Example Equations

For a vertical, fixed-roof storage tank the following equations apply:[1]

$$L_T = L_S + L_W$$

where: $L_S = 365\ W_V V_V K_E K_S$

$L_W = 0.0010\ M_V P_{VA} Q K_N K_P$

L_T = total loss, lb/yr

L_S = standing storage loss, lb/yr

L_W = working loss, lb/yr

V_V = tank vapor space volume, ft^3

$$V_V = \frac{\Pi}{4 D^2 H_{VO}}$$

W_V = vapor density, lb/ft^3

$$W_V = \frac{M_V P\ VA}{RT_{LA}}$$

K_E = vapor space expansion factor, dimensionless

$$K_E = \frac{\Delta T_V}{T_{LA}} + \frac{\Delta P_V - \Delta P_B}{P_A - P_{VA}}$$

K_S = vented vapor space saturation factor, dimensionless

$$K_S = \frac{1}{1 + 0.053 P_{VA} H_{VO}}$$

D = diameter, ft

H_{VO} = vapor space outage, ft

M_V = molecular weight of vapor, lb/lb-mole

P_{VA} = vapor pressure at the daily average liquid surface temperature, psia

T_{LA} = daily average liquid surface temperature, °R

ΔT_V = daily vapor temperature range, °R

ΔP_V = daily vapor pressure range, psia

ΔP_B = breather vent pressure setting range, psi

P_A = atmospheric pressure, psia

Q = annual net throughput, lb/yr

K_N = working loss turnover factor, dimensionless

K_P = working loss product factor, dimensionless

Applicability Considerations:

- Equations and tables that further define these equation variables are presented in Supplement E of AP-42, "Storage of Organic Liquids."
- Emissions from horizontal tanks can be calculated by making adjustments to parameters in the fixed-roof equations. There are step-wise examples regarding how to proceed with adjustments in Supplement E of AP-42, "Storage of Organic Liquids."
- Supplement E also supplies the vapor pressure equation constants, molecular weight, boiling point at 1 atmosphere, liquid density at 60°F, and vapor pressure at varying temperatures for a variety of compounds (including MC).

SECTION 2 - EQUIPMENT LEAKS EXAMPLE

The following equipment leaks example is presented for estimating equipment leaks emissions based on VOC emission factors. This example presents the simplest of five methods delineated in the "Protocols" document, and should only be employed when other data are not available.[2] The purpose of its presentation is to alert the reader to the site-specific and chemical property considerations that are required for the estimation of MC emissions from equipment leaks. The estimation methodology is based on the following general points:

- Annual emission rates from equipment leaks in this example are based on the VOC emission factors (kg of VOC/hr) presented in Table A-1.[2]

- In applying the emission factors to equipment leaks, MC can be considered as a light liquid VOC because its vapor pressure is greater than 0.3 kPa. It can be assumed that MC is emitted like other VOC compounds.

- The annual MC emission rate from each equipment component type is the product of the appropriate emission factor, the percent MC handled by the equipment component, the maximum number of hours the equipment handles MC-laden material, and the number of equipment components that come in contact with the MC. The emission rates for each type of equipment component (i.e., pump seals, compressor seals, flanges, etc.) are summed to obtain a total equipment leak emission rate for each facility.

Sample Calculation

Example Site-Specific Information

- Three mechanical pump seals between mixing tank and filling lines. All are within a building.

  3 lines - 3 pump seals

- Hours of handling

  3 lines - 8 hr/day, 5 day/wk

- Amount of MC consumed = 496,944 lb/yr. Includes consumption for products and line flush.

- Amount of MC-based products packaged = 2,828,797 lb/yr

TABLE A-1. AVERAGE EMISSION FACTORS FOR PROCESS EQUIPMENT COMPONENTS USED TO ESTIMATE VOC EMISSIONS FROM EQUIPMENT LEAKS

Equipment Component (Emission Source)	Emission Factor[a,b] (kg VOC/hr-component)
Pumps - Liquid[b]	0.494
Compressors	0.228
Flanges	0.00083
Valves - Gas	0.0056
- Liquid[b]	0.0071
- Heavy Liquid	0.00023
Sampling Connections	0.0150
Open-Ended Lines	0.0017

[a] Emission factors were developed for VOC emitted from equipment components used in the synthetic chemical manufacturing industry (SOCMI).

[b] Liquid refers to light liquid and is defined as a petroleum liquid with a vapor pressure greater than the vapor pressure of kerosene.

Source: Reference 2.

Calculations

1. Hours of Handling = 8 hr/day · 5 day/wk · 52 wk/yr = 2,080 hr/yr
   --assumes 52 wk/yr

2. Percent MC handled by the equipment components

$$\frac{496{,}944 \text{ lb MC/yr}}{2{,}828{,}797 \text{ lb MC-based product/yr}} \cdot 100 = 17.6 \text{ weight-percent}$$

   - assumes the amount of MC used for cleanup and line flush is handled by the same component
   - assumes concentration of MC is similar for all lines

3. Number of and emission factors for equipment components.

   Number and type of equipment component = three mechanical pump seals.
   Emission factor for pump seals = 0.494 kg MC/hr

4. Annual MC Emission Rate = $\frac{\text{Number of Components}}{}$. $\frac{\text{Weight-percent of MC Handled}}{}$ · Hours of Handling · Emission Factor

$$= (3)\ (0.176)\ (2{,}080 \text{ hr/yr})\ (0.0494 \text{ kg MC/hr}) \cdot \frac{1 \text{ Mg}}{1{,}000 \text{ kg}}$$

$$= 0.05 \text{ Mg MC/yr}$$

SECTION 3 - SECONDARY WASTE STREAM EXAMPLE

The following secondary waste stream wastewater example calculation is presented as an example method for the estimation of MC emissions from wastewater streams. For a more detailed and accurate estimation methodology for MC emissions from wastewater streams, the reader is referred to the "Industrial Wastewater Volatile Organic Compound Emissions -- Background Information for Proposed Standards" document.[3] The purpose of this presentation is to alert the reader to some of the site-specific considerations required for the estimation of MC emissions from wastewater streams.

Example Method

- Secondary waste streams include wastewater streams, and organic-liquid or solid wastes.

- Emissions from wastewater streams can be calculated from the MC concentration, and the daily influent wastewater flowrate. In order to develop maximum emission estimates, it could be assumed that all of the MC in the wastewater is released to the atmosphere. In this example, it is assumed wastewater is generated 260 days per year.

$$\text{MC Emission Rate [Mg MC/yr]} = \frac{\text{Influent Wastewater Flowrate to Treatment or Storage [gal/day]} \cdot \text{Influent Concentration [}\mu\text{g MC/}\ell\text{]} \cdot [3.79\ \ell/\text{gal}]}{\frac{10^{12}\ \mu\text{g}}{1\ \text{Mg}} \cdot \frac{\text{yr}}{260\ \text{day}}}$$

- Emissions from organic-liquid or solid wastes can be calculated from the daily volume of waste generated and the weight percent of MC.

$$\text{Emission Rate [Mg MC/Yr] from Liquid or Solid Wastes} = \text{Daily Volume [gal/day]} \cdot \text{Density MC [Mg/gal]} \cdot \frac{260\ \text{days}}{\text{yr}} \cdot \text{MC Composition (wt-percent)}$$

REFERENCES

1. Compilation of Air Pollutant Emission Factors, Volume I: Stationary Point and Area Sources, Fourth Edition, AP-42, Supplement E, U.S. Environmental Protection Agency, Research Triangle Park, NC.

2. Protocols for Generating Unit-Specific Emission Estimates for Equipment Leaks of VOC and VHAP, EPA-450/3-88-010, U.S. Environmental Protection Agency, Research Triangle Park, NC, 1988.

3. Industrial Wastewater Volatile Organic Compound Emissions -- Background Information for BACT/LAER Determinations, EPA-450/3-90-004, March 1990, Chapter 4.

APPENDIX B

NATIONAL LIST OF PAINT STRIPPER FORMULATORS (JULY 1987)

NATIONAL LIST OF PAINT STRIPPER FORMULATORS (JULY 1987)

Company Name	Location
Alvin Products, Inc.	Worcester, MA
Amchem Products	Ambler, PA
American Niagara Corp.	Atlanta, GA
Ashland Chemical Co.	Columbus, OH
A-Z Midwest Factory Supply	Addison, IL
Beaver Alkali Products	Rochester, PA
Beck Chemicals, Inc.	Cleveland, OH
Behlen, H. & Bros., Inc.	Amsterdam, NY
Benco	Fontana, CA
Benco	Crossville, TN
Besway Chemical Systems, Inc.	Madison, TN
Cedtex Chemical	Southfield, MI
Certified Coating Pdts., Inc.	Los Angeles, CA
Chemclean Corporation	College Point, NY
Chemical Products Co., Inc.	Aberdeen, MD
Chemical Systems Research	West Bloomfield, MI
Consolidated Chemical	St. Louis, MO
Dap, Inc.	Tipp City, OH
Day, James B., & Co.	Carpentersville, IL
Delta Foremost Chemical Corp.	Memphis, TN
Diversey Wyandotte Corp.	Wyandotte, MI
Dober Chemical Corp.	Midlothian, IL
Dynaloy, Inc.	Hanover, NJ
Dytex Chemical Co., Inc.	Central Falls, RI
Ecco Chemicals, Inc.	Dallas, TX
Eldorado Chemicals	San Antonio, TX
Ensign Products	Cleveland, OH
Enterprise Co.	Wheeling, IL
Excelda Mfg. Co.	Ferndale, MI
Excelsior Varnish	Cleveland, OH
EZE Products	Greenville, SC

NATIONAL LIST OF PAINT STRIPPER FORMULATORS (JULY 1987) (Continued)

Company Name	Location
Formby's Inc.	Olive Branch, MS
FPNS	Henderson, KY
Fuller-O'Brien Paints	South San Francisco, CA
Gage Products	Ferndale, MI
Goodrich Products	Highlands, TX
Grow Group, Inc.	Detroit, MI
Grow Group, 2 (Nat'l Aerosol Pdts)	Los Angeles, CA
Harley Chemicals	Camden, NJ
Hexcel Chemicals	Lodi, NJ
Higley Chemicals	Dubuque, IA
Hillyard Chemicals	St. Joseph, MO
International Chemicals	Philadelphia, PA
International Paints	Union, NJ
International Paints	Houston, TX
Isochem Products	Lincoln, RI
J & S Chemicals	Macedonia, OH
Jasco Chemicals	Mountain View, CA
KCI Chemicals	La Porte, IN
Kerns United	Calumet City, IL
Key Chemicals	Philadelphia, PA
Klean Strip	Memphis, TN
Kwick Kleen Industries	Vincennes, IN
London Chemicals	Bensenville, IL
Lowe Brothers Co.	Cleveland, OH
Madison Bionics	Oak Brook, IL
Magnuson Products	Clifton, NJ
Man-Gill Co.	Cleveland, OH
Mantrose-Hauser Company	Attleboro, MA
McGean Rohco Inc.	Los Angeles, CA
Midland Lab	Des Moines, IA
Mitchell-Bradford Chemicals	Milford, CT

NATIONAL LIST OF PAINT STRIPPER FORMULATORS (JULY 1987) (Continued)

Company Name	Location
National Solvent Corp.	Medina, OH
NCH Corp.	Irving, TX
Nuvite Chemicals	Brooklyn, NY
Oakite Products	Metuchen, NJ
Oakite Products	City of Industry, CA
Oakite Products	Romulus, MI
Oakite Products	Houston, TX
Oxford Chemicals	Atlanta, GA
Par-Chem Products	Houston, TX
PBNS	Henderson, KY
Penetone Corporation	Tenafly, NJ
Pennwalt Corporation	Carson, CA
Pennwalt Corporation	Marion, OH
Petrocon Marine & Ind.	Brooklyn, NY
Pioneer Chemicals	West Point, OH
Prillaman Company	Martinsville, VA
Product-Sol Inc.	Birmingham, MI
Proko Industries	Dallas, TX
PyRock Chemicals	Long Island City, NY
Rap Products	Bay City, MI
Red Devil Paints	Mount Vernon, NY
Reliable Paste and Chemicals	Chicago, IL
Reliable Remover and Lacquer	Irvington, NJ
Savogran Company	Norwood, MA
Savogran Company	Addison, IL
Savogran Company	Los Angeles, CA
Sermac Industries	Altoona, PA
Sheldahl	Northfield, MN
Sherwin-Williams	Richmond, KY
Sherwin-Williams	Chicago, IL
Staples, H.F.	Merrimack, NH

NATIONAL LIST OF PAINT STRIPPER FORMULATORS (JULY 1987) (Concluded)

Company Name	Location
Star Bronze	Alliance, OH
Sterling-Clark-Lurton	Malden, MA
Strip-Tech	Hendersonville, TN
Stripping Products (BIX)	Old Hickory, TN
Stuart Radiator	Merced, CA
Sunshine Chemicals	West Hartford, CN
Texas Refinery	Fort Worth, TX
Texo Corp	Cincinnati, OH
Tower Chemicals	Palmer, PA
Tropical Industrial Coatings	Brunswick, OH
United Gilsonite	Scranton, PA
Urban Chemicals	Deerfield, IL
Watson-Standard	Harwick, PA
William M. Barr & Co.	Memphis, TN
Wilson-Imperial	Newark, NJ
Zep Manufacturing	Atlanta, GA

Source: Memorandum from E. Moretti, Radian Corporation, to the U.S. Environmental Protection Agency. Emission Estimates and Controls for Emissions from Paint Stripper Formulation Facilities (DRAFT). September 11, 1987.

APPENDIX C

NATIONAL LIST OF AEROSOL PACKAGERS

NATIONAL LIST OF AEROSOL PACKAGERS (JULY 1987)

	COMPANY	CITY	STATE	LATITUDE	LONGITUDE	PRODUCT CODE	REFERENCE
1	ACCRA PAC GROUP	ELKHARDT	IN	414100	855800	A,C,D,F,K	1,3,4
2	ADTECH DESIGN CO.	HORSHAM	PA	401100	750800	A,B,C,D,E,F,H,I	3
3	ADVANCE AEROSOL & CHEMICAL CO.	GENOA CITY	WI	423000	882000	A,C,D,E,F,K	3,6
4	AEROSOL RESEARCH LAB, INC.	HOLBROOK	MA	420900	710100	C,D,E,G	3
5	AEROSOL SERVICES CO., INC.	CITY OF INDUSTRY	CA	340200	1175600	A,B,C,D,G,H,K	3,4,5
6	AEROSOL SYSTEMS	MACEDONIA	OH	411900	813100	A,C,D,E,F	3,9
7	AEROSOL WEST	VENTURA	CA	341700	1191800	A,C,D,E,H	3
8	AEROTECH INDUSTRIES	GARLAND	TX	333000	943700	A,D,F,K	3
9	AERO-KING, INC.	BRONX	NY	405100	735400	C,D,E	3
10	AERVOE PACIFIC CO.	SAN LEANDRO	CA	374400	1220900	A,D,F,K	4,9
11	AIROSOL CO., INC.	NEODESHA	KS	372500	954100	A,D,C,E,G,I,K	3
12	ALTAWOOD INC.	GARDENA	CA	335300	1181800		5
13	AMERICAN AEROSOLS	HOLLAND	MI	424700	860700	A,D	3
14	AMERICAN NIAGARA CORP.	ATLANTA	GA	334500	842300		6
15	AMREP, INC.	MARIETTA	GA	335700	843300	A,C,D,E,K	3,6,9
16	AMWAY CORP.	ADA	MI	425900	853000		5
17	APOLLO INDUSTRIES, INC.	SMYRNA	GA	335300	843100		5
18	ATI, INC-1	TOTOWA	NJ	405400	741300		1,9
19	ATI, INC-2 (ARMSTRONG LABORATORIES DIV.)	WEST ROXBURY	MA	421700	710900	C,D,E,G,H	2,3,9
20	BARR CO.	NILES	IL	420200	874800	A,C,D,G,I	2,9
21	BARRETT CHEMICAL CO.	PHILADELPHIA	PA	395700	751000	A,D	3
22	BEECHAM HOME IMPROVEMENT	CLIFTON	NJ	405200	740900		5
23	BISSELL-1 (PENN CHAMP, INC)	EAST BUTLER	PA	403300	795100	A,C,G	8
24	BISSELL-2 (PENN CHAMP, INC)	RIVERSIDE	CA	335900	1172200	A,C,G	3,4
25	BORDEN (KRYLON DEPARTMENT)	COLUMBUS	OH	395800	830000	F	7
26	BOXAL METALS, INC.	FAIRLAWN	NJ	405600	740800		4
27	BOYLE-MIDWAY (AMERICAN HOME PRODUCTS)	CRANFORD	NJ	404000	741800		5,6
28	BRISTOL-MEYERS-1 (CLAIROL)	NEW YORK	NY	404500	735700	G	5
29	BRISTOL-MEYERS-2 (LENK CO.)	FRANKLIN	KY	364300	863500		1,9
30	BROUGHTON FOODS CO.	MARIETTA	OH	392500	812700		5
31	B&B AEROSOL PACKAGING CO., INC.	CHICAGO	IL	415200	873900	A,C,D,E,F,G,K	3
32	CAMIE-CAMPBELL, INC.	ST. LOUIS	MO	383700	901200		6,9
33	CARE LABORATORIES, INC.	COLLEGEVILLE	PA	401100	752800	A,D,E,F	3
34	CARSON CHEMICALS, INC.	NEW CASTLE	IN	395500	852200		5
35	CARTER-WALLACE, INC.	CRANBURY	NJ	401900	743100		5
36	CASE-MASON FILLING, INC.	JOPPA	MD	392600	762200	C,D,G,H,K	3,9
37	CHAMPION LABS (PYROIL CO.)	ALBION	IL	382300	880400	A,D	3,5
38	CHASE PRODUCTS CO	MAYWOOD	IL	415300	875100		4,9
39	CHEM TECH, INC.	WILMINGTON	DE	394500	751400	A,D	5

NATIONAL LIST OF AEROSOL PACKAGERS (JULY 1987)

	COMPANY	CITY	STATE	LATITUDE	LONGITUDE	PRODUCT CODE	REFERENCE
40	CHEMSCOPE CORP.	ARLINGTON	TX	324400	970700	A,B,D,F	3,9
41	CHEMSICO, INC.	ST. LOUIS	MO	383700	901200	A,C,D,E,F,I	3,9
42	CHEMSPRAY, INC.	TOTOWA	NJ	405400	741300	A,C,D,G,H,I K	3
43	CHEMTRONICS	HAUPPAGE	NY	405000	731200	A,D,E,G,H	3,9
44	CHEM-PACKAGING CORP.	POMPANO BEACH	FL	261400	800800	A,C,D,E	3
45	CHEM-PAK, INC.	WINCHESTER	VA	391100	781000	A,B,D,F	3,6,9
46	CHEM-TECH, LTD.	DES MOINES	IA	413500	933700	E,I	3
47	CHESEBROUGH-PONDS, INC.	GREENWICH	CT				5
48	CLAPP INDUSTRIES CORP.	POTOMAC	IL	401800	874800	A,B,C,D,G,K	3
49	CLAYSON, INC. (GEM, INC.)	BYHALIA	MS	345200	894100	C,D,G	1,3,8,9
50	CLINE-BUCKNER, INC.	ARTESIA	CA	235200	1180500	C,D,E	3
51	CONNECTICUT AEROSOLS, INC.	MILFORD	CT	411400	730400		3,9
52	CONTACT INDUSTRIES, INC.	ELIZABETH	NJ	404000	741300	A,C,D,E,F,G,H,I	3,4,9
53	CONTINENTAL LABORATORIES, INC.	MADRID	IA	415300	934900	A,C,D,E,G,H,I	3
54	CPC-1 (PETERSON/PURITAN, INC.)	CUMBERLAND	RI	415700	712500	C,G	6
55	CPC-2 (PETERSON/PURITAN, INC.)	DANVILLE	IL	400800	873700	A,B,C,D,E,G,H,I,K	4,6
56	CPC-3 (PETERSON/PURITAN, INC.)	ENGLEWOOD CLIFFS	NJ	405300	735700		6
57	CRC CHEMICALS	WARMINSTER	PA	401200	750600		6
58	CREATIVE PRODUCTS CORP.	NEW HAVEN	CT	411800	755500	A,B,C,D,G,H,K	3
59	CROWN INDUSTRIAL PRODUCTS CO.	HEBRON	IL	422800	882600	D,F	3,6,9
60	CUSTOM-PAK PRODUCTS, INC.	GERMANTOWN	WI	431400	880600	D,F	3,5
61	C.S.A. LIMITED, INC.	HOUSTON	TX	294600	952200	A,C,D,E,G I	3
62	DELCOR LABORATORIES, INC.	E. STROUDSBERG	PA	410000	751100	G,H	3
63	DEMERT & DOUGHERTY, INC.	OAK BROOK	IL	415100	875800	A,C,D,G,K	1,3,9
64	DESOTO, INC.	DES PLAINES	IL	420300	875200		5
65	DEVOE & RAYNOLDS	LOUISVILLE	KY	381500	854600		5
66	DYMON, INC.	KANSAS CITY	KS	390700	943800	A,D,E	3,5
67	EMBALMERS SUPPLY CO.	WESTPORT	CT	410900	732200		5
68	ENTERPRISE SALES CO.	LOS ANGELES	CA	340400	1181500	D,E,K	3
69	ESKA PACKAGING CORP.	LAKEWOOD	NJ	400600	741300	A,D,E	3
70	EVEREADY PRODUCTS CORP.	CLEVELAND	OH	413000	814200	A,C,D,F,K	3,5
71	E.J. MCKERNAN CO.	GARDEN GROVE	CA	334700	1175500		4,8
72	FASSE PAINT CO., INC.	SHEBOYGAN FALLS	WI	434200	874500	F	3
73	FAULTLESS STARCH (BON AMI CO.)	KANSAS CITY	MO	390600	943500		5
74	FLUID PACKAGING CO., INC.	LAKEWOOD	NJ	400600	741300	G,H	4,9
75	FORREST PAINT CO.	EUGENE	OR	440500	1230400		5
76	FRANK ORLANDI, INC.	FLUSHING	NY	404500	734900	C,D,K	3
77	FRANKLIN OIL CORP.(RICOL DIV.)	CLEVELAND	OH	413000	814200	D	6,9
78	FULTON CO.	SUMTER	SC	335500	802100	F	3

NATIONAL LIST OF AEROSOL PACKAGERS (JULY 1987)

	COMPANY	CITY	STATE	LATITUDE	LONGITUDE	PRODUCT CODE	REFERENCE
80	GEMINI AEROSOLS, INC.	EL RENO	OK	353200	975700	A,F,	3
81	GEMINI LACQUERS	BEVERLY	MA	423300	705300		5
82	GEM, INC.	BYHALIA	MS	345200	884100	C,D,G	3
83	GENERAL PAINT	CARY	IL	421300	881400		5
84	GEORGIA PACIFIC CORP.	ATLANTA	GA	334500	842300		5
85	GILETTE CO.	ANDOVER	MA	424000	710800		6
86	GLIDDEN PAINTS	CLEVELAND	OH	413000	814200		6
87	GROW GROUP-1 (CELLO CORP.)	SANTA FE SPRINGS	CA	335700	1180400		5
88	GROW GROUP-2 (NATIONAL AEROSOL PRODUCTS)	CERRITOS	CA	335200	1180500	A,C,D,E,F,G,I,K	2,7,9
89	GUARDSMAN (AMERICAN AEROSOL INC.)	HOLLAND	MI	424700	860700		1,3,5,9
90	GUEST PACKAGING	RAHWAY	NJ	403700	741600	C,G	3
91	HOLT LLOYD (LPS PRODUCTS)	TUCKER	GA	335100	841300	D	6,7
92	HOWARD JOHNSON	NORTH QUINCY	MA	421700	710100		5
93	HYDROSOL	BURR RIDGE	IL	414700	875500	A,C,D,E,F,K	3,5
94	HYSAN CORP.	CHICAGO	IL	415200	873900	A,C,D,E,F,G,K	3
95	IKI MFG. CO.	EDGERTON	WI	425000	890400	A,C,D,E,G,I,H,K	1,3,9
96	ILLINOIS BRONZE PAINT CO.	LAKE ZURICH	IL	411200	880500	A,D,F,K	1,3,9
97	JET-AER CORP.	PATERSON	NJ	405500	741100	A,C,D,E,I,K	4
98	JL MFG.	GRANBURY	TX	332700	974700		3
99	KNIGHT OIL CORP.	JOHNSTOWN	NY	430000	742200	A,C,D,F	3,9
100	KOWALRAD PRODUCTS, INC.	PANDORA	OH	405700	835800	A,C,D,F	3
101	LAMAUR COMPANY	MINNEAPOLIS	MN	445900	931600		6
102	LAWRENCE-MCFADDEN CO.	PHILADELPHIA	PA	395700	751000		7,9
103	LAWSON CHEMICAL PRODUCTS CO.	TORRANCE	CA	335000	1181900	A,D,E,F	3
104	MARCY LABS	ADDISON	IL	415600	875900	A,G,I,K	1,3,9
105	MBL INDUSTRIES, INC.	SANTA ANA	CA	334600	1175200	A,C,D,E,K	3
106	METROVAL, INC. (EPIC INDUSTRIES, INC.)	METUCHEN	NJ	403200	742200		5,9
107	MIDCO PRODUCTS, CO	MARYLAND HEIGHTS	MO	384300	902600	A,C,D,E,F,I	3,9
108	MOBILE PAINT	THEODORE	AL	303300	881000		5
109	MOHAWK FINISHING	AMSTERDAM	NY	424600	741100		5
110	MOHAWK LABORATORIES	IRVING	TX	324900	965600		5
111	MORTON PHARMACEUTICALS, INC.	MEMPHIS	TN	350800	900300	A,C,D,F,I	3,9
112	NATIONWIDE INDUSTRIES	PANDORA	OH	405700	835800		8
113	NEW ENGLAND AEROSOL & PACKAGING CORP.	WOBURN	MA	422900	710900	A,C,D,E,F,G,K,I	3,5
114	NEW YORK BRONZE-1	ELIZABETH	NJ	404000	741300		5,6
115	NEW YORK BRONZE-2	TAYLOR	PA	412200	754300	F	6
116	NORTHEAST PACKAGING INC.	ORANGE	CT	411700	730200	A,C,D,E,F,G,H	3
117	NOXELL, INC.	BALTIMORE	MD	391700	763700		5
118	OAKITE PRODUCTS-1 (CLAIRE MFG. CO.)	ADDISON	IL	415600	875900	A,C,D,E	1,5,6,9

NATIONAL LIST OF AEROSOL PACKAGERS (JULY 1987)

	COMPANY	CITY	STATE	LATITUDE	LONGITUDE	PRODUCT CODE	REFERENCE
120	OAKITE PRODUCTS-3 (SPRAYWAY INC.)	ADDISON	IL	415600	875900		2,6,9
121	ORB INDUSTRIES, INC.	UPLAND	PA	395100	752300	D,E,F,K	3,9
122	PACTRA INDUSTRIES, INC.	UPLAND	CA	340600	1173900	F	7
123	PEERLESS TUBE CO.	BLOOMFIELD	NJ	404800	741200		4
124	PEL ASSOCIATES	NORTH BRANCH	NJ	403600	744100	A,C,D,G,H,K	3,5,9
125	PERCY HARMS CORP.	WHEELING	IL	420800	875500	F	3,6
126	PHARMASOL CORP.	RANDOLPH	MA	421000	710200	A,C,D,G,H,I,K	3
127	PIEDMONT LABS, INC.	GAINESVILLE	GA	341800	835000		3,5
128	PLASTI-KOTE CO.	MEDINA	OH	410800	815200	A,F	3,6
129	PLAZE AEROSOLS, INC.	ST. LOUIS	MO	383700	901200	A,C,D,E,F,G,I,J	4,9
130	PLOUGH, INC.	MEMPHIS	TN	350800	900300		5
131	PRESSPAK, INC.	SUGARLAND	TX	293700	953800	G,H	3
132	PRICE DRISCOLL CORP.	FARMINGDALE	NY	404400	732700		6,9
133	QUEST PACKAGING INC.	HOUSTON	TX	294600	952200		3
134	RAABE PAINT CO., INC.	WAUWATOSA	WI	430300	880000	A,F,D	3,5
135	RADIATOR SPECIALTY CO.	CHARLOTTE	NC	351300	805100		5
136	RALPH SHRADER	DETROIT	MI	422000	830300		5
137	RANDOLPH PRODUCTS CO.	CARLSTADT	NJ	405000	740600	F	3,4
138	RAWN COMPANY, INC.	SPOONER	WI	455000	915300		5
139	REALEX CORP.	BATON ROUGE	LA	302700	911100		5
140	REMPAK INDUSTRIES, INC.	CARTERET	NJ	403400	741300	A,C,D,E,G,H,I	3
141	RITE OFF, INC.	BAY SHORE	NY	404300	731500	A,C,D,E,K	4
142	RUDD PAINT & VARNISH CO.	SEATTLE	WA	473600	1222000		5
143	RUSTOLEUM CORP.	MUNDELEIN	IL	421600	880000		5
144	SC JOHNSON	RACINE	WI	424400	874800		6
145	SCHERING CORP.	ST. LOUIS	MO	383700	901200	H	3
146	SCOTTS LIQUID GOLD	DENVER	CO	394400	1045900		5
147	SECURE AMERICA CORP.	MARIETTA	GA	335700	843300	K	3
148	SECURITY EQUIPMENT CORP.	ST. LOUIS	MO	383700	901200	A,C,D,K,E	3
149	SEYMOUR OF SYCAMORE	SYCAMORE	IL	415900	884100		6
150	SHERWIN WILLIAMS-1 (DUPLI-COLOR)	ELK GROVE VILLAGE	IL	420100	875900		6
151	SHERWIN WILLIAMS-2 (SPRAYON PRODUCTS)	ANAHEIM	CA	335000	1175500		6
152	SHERWIN WILLIAMS-3 (SPRAYON PRODUCTS)	BEDFORD HEIGHTS	OH	412300	813000		2,6
153	SHIELD PACKAGING CO.	CANTON	MA	420900	710900	A,C,D,E,F,G,I,K	3
154	SHIELD PACKAGING OF CALIF.	CHINO	CA	340100	1174100	A,C,D,E,G,K,I	3
155	SHULTON, INC.	WAYNE	NJ				5
156	SOUTHLAND PAINT CO.	GAINESVILLE	TX	333800	970800	F	7
157	SPEER PRODUCTS-1 (PET CHEMICALS)	MIAMI SPRINGS	FL	254900	801800		5
158	SPEER PRODUCTS-2	MEMPHIS	TN	350800	900100	A,C,D,E,K,I	3,8

NATIONAL LIST OF AEROSOL PACKAGERS (JULY 1987)

	COMPANY	CITY	STATE	LATITUDE	LONGITUDE	PRODUCT CODE	REFERENCE
160	SPRAY PRODUCTS CORP.	NORRISTOWN	PA	400700	752100	A,D,E,F	3,9
161	SSP CHEMICAL CO., INC.	ROYERSFORD	PA	401100	753300	C,D,E,F	3
162	STALFORT CONSUMER PRODUCTS	HAVRE DE GRACE	MD	393300	760600	A,C,D,E	3
163	STANDARD BRANDS PAINT CO.	TORRANCE	CA	335000	1181900		1,9
164	STANHOME, INC.	WESTFIELD	MA	420700	724500	A,C,D,E,G	2,3,8,9
165	STAR CHEMICAL CO., INC.	HINSDALE	IL	414800	875600	F,D	3,5
166	STONERS INK CO.	QUARRYVILLE	PA	395400	761000		5
167	STROBEL PRODUCTS, INC.	LOUISVILLE	KY	381500	854600	A,C,D,F	3,9
168	SUN LABS-1 (SOUTHEAST PACKAGING CO.)	ATLANTA	GA	334500	842300	A,C,E,G,K	4,5
169	SUN LABS-2	CHATSWORTH	CA	341500	1183600	G	3,5
170	TALLEY INDUSTRIES (TIME-MIST INC.)	WATERBURY	CT	413300	730300	E	2,9
171	TECH LUBE	ISLIP	NY	404400	731300		6,9
172	TECHNAIR LABS, INC.	RAHWAY	NJ	403700	741600		4
173	TESTOR CORP	ROCKFORD	IL	411500	875800		5
174	THEOCHEM LABORATORIES, INC.	TAMPA	FL	275700	822700		8
175	3M CO.	ST. PAUL	MN	445700	930600	A,C,D,H	3,6
176	TRIMARK PACKAGING, INC.	ENNIS	TX	322000	963800		4
177	ULTRAMOTIVE CORP	BETHEL	VT	435000	723800	A,C,D,G,K,I	3,8
178	UNCOMMON CONGLOMERATES	ST. PAUL	MN	445700	930600		6
179	UNIPACK, INC.	PITTSBURGH	PA	402600	800100	A,C,D,E,F,G,H,K	3
180	UNIVERSAL SPECIALTY PACKAGING	ROSELLE	NJ	404000	741500	B,G,H,K	3,5
181	US AVIEX CO.	NILES	MI	415000	861500	A,C,D,E	3
182	US PACKAGING CORP	WHEELING	IL	420800	875500	A,C,D,F,K	3
183	VALJEAN CORP.	INDIAN HARBOUR BEACH	FL	280900	803600	C,E,G,H,I	3
184	VICTOR INDUSTRIES CORP	CHICO	CA	394400	1215000	A,B,C,D,E,F,G,H,I,K	3
185	WHITMIRE RESEARCH	ST. LOUIS	MO	383700	901200		6
186	WILLIAM BARR & CO.	MEMPHIS	TN	350800	900300		3
187	ZEP MANUFACTURING CO.	ATLANTA	GA	334500	842300	D,E	5,6,9
188	ZIP AEROSOL PRODUCTS	CANOGA PARK	CA	341200	1183500		6,9
189	ZOE CHEMICAL CO., INC.	NEW HYDE PARK	NY	404400	734100	C,D,E,I	3,4,8
190	ZYNOLYTE PRODUCTS CO.	COMPTON	CA	335400	1181300		7

NATIONAL LIST OF AEROSOL PACKAGERS (JULY 1987)

	COMPANY	CITY	STATE	LATITUDE	LONGITUDE	PRODUCT CODE	REFERENCE
PACKAGING FACILITIES WITH UNIDENTIFIED LOCATION							
191	AEROSOL SPECIALTIES						5
192	AMERICAN JET-WAY						5
193	CESSCO, INC.						5
194	CHEMICAL PACKAGING SERVICES, INC.						5
195	CHEMI-COATINGS, INC.						5
196	C.F. BURGER						5
197	EMKO CO.						5
198	FO-MO PRODUCTS						5
199	IG-LO PRODUCTS CORP.						5
200	INTERDYNAMICS, INC.						5
201	JIM YORK CHEMICALS CO.						5
202	MAGNAFLUX CORP.						5
203	MAJOR PAINT						5
204	MILLER-STEPHENSON						5
205	NELSON PAINT						5
206	PETRO CHEMICAL PRODUCTS						5
207	SHIELD CHEMICAL CO.						5
208	SHIRLO, INC.						5
209	TECH SPRAY						5
210	TECHNICAL CHEMICALS						5
211	TRADCO						5
212	UNITED COATINGS						5

PRODUCT CODE:

A = AUTOMOTIVE
B = FOOD
C = HOUSEHOLD
D = INDUSTRIAL
E = INSECTICIDE
F = PAINT
G = PERSONAL
H = PHARMACEUTICAL
I = VETERINARY
K = SPECIALTY

REFERENCE:

1 = STANDARD AND POOR'S REGISTER, 1980
2 = CORPORATE AFFILIATIONS, 1986
3 = AEROSOL AGE BUYER'S GUIDE, OCTOBER 1986
4 = AEROSOL AGE (1985 ISSUES)
5 = CSMA PRESSURIZED PRODUCTS SURVEY, 1985
6 = EPA MEMORANDUM, DOCUMENTATION OF TELEPHONE COMMUNICATIONS CONCERNING THE PACKAGING AND LOCALIZED LARGE-SCALE USERS OF AEROSOL PRODUCTS, PREPARED BY GARY BOCKOL, RADIAN CORPORATION, FEBRUARY 6,1987
7 = OCCUPATIONAL EXPOSURE AND ENVIRONMENTAL ASSESSMENT OF FOUR CHLORINATED SOLVENTS WHEN USED IN AEROSOLS, MARCH 28,1986
8 = CSMA VENDORS TO THE TRADE GUIDE, 1986
9 = THOMAS REGISTER, 1895

APPENDIX D

TRIS LIST OF PHARMACEUTICAL FACILITIES USING METHYLENE CHLORIDE

Treatments for Methylene Chloride (CAS 000075092)

SIC	TRIS ID	FACILITY NAME	WC	WTC	IC	STI	EFFIC	OD
283	00617BBTTCROADN	ABBOTT CHEMICALS INC.	A	P21	3		08500	N
283	00617BBTTCROADN	ABBOTT CHEMICALS INC.	A	B11	3		09500	N
283	00617BBTTCROADN	ABBOTT CHEMICALS INC.	A	A02	1		09800	N
283	00617MRCKSSTATE	MERCK SHARP & DOHME QUIMICA DE P.R.	L	F01	1	Y	00000	
283	00617MRCKSSTATE	MERCK SHARP & DOHME QUIMICA DE P.R.	A	A03		Y	10000	Y
283	00617MRCKSSTATE	MERCK SHARP & DOHME QUIMICA DE P.R.	A	A02	1		08000	N
283	00617MRCKSSTATE	MERCK SHARP & DOHME QUIMICA DE P.R.	A	A02	1		07100	N
283	00617MRCKSSTATE	MERCK SHARP & DOHME QUIMICA DE P.R.	A	A02	1		07250	N
283	00617PFZRPHIGHW	PFIZER PHARMACEUTICALS INC.	W	C11	2		00000	N
283	00617PFZRPHIGHW	PFIZER PHARMACEUTICALS INC.	A	A03	2		07300	N
283	00617PFZRPHIGHW	PFIZER PHARMACEUTICALS INC.	L	F11	2		10000	Y
283	00617PFZRPHIGHW	PFIZER PHARMACEUTICALS INC.	A	A02	2		10000	Y
283	00617THPJHHIGHW	UPJOHN MFG. CO.	A	A03	2		02000	N
283	00628LLLLY65THI	ELI LILLY INDUSTRIES INC.	A	P11	2		05400	Y
283	00628LLLLY65THI	ELI LILLY INDUSTRIES INC.		NA	NA		NA	N
283	00661SQBBMSTATE	SQUIBB MFG. INC.	A	A02	1		09000	N
283	00661SQBBMSTATE	SQUIBB MFG. INC.	A	A02	1		09500	N
283	00661SQBBMSTATE	SQUIBB MFG. INC.	A	A03	1		09500	N
283	00661SQBBMSTATE	SQUIBB MFG. INC.	A	A04	1		09500	N
283	00661SQBBMSTATE	SQUIBB MFG. INC.	A	A02	1	Y	00000	
283	00661SQBBMSTATE	SQUIBB MFG. INC.	A	A03		Y	09400	N
283	00661SQBBMSTATE	SQUIBB MFG. INC.	W	F01	3		10000	N
283	00661SQBBMSTATE	SQUIBB MFG. INC.	L	F01	1		10000	N
283	00661SQBBMSTATE	SQUIBB MFG. INC.	W	B11	3		09800	N
283	00661SQBBMSTATE	SQUIBB MFG. INC.	L	R11	1		09000	N
283	00661SQBBMSTATE	SQUIBB MFG. INC.	A	A02	1		09000	N
283	00661SQBBMSTATE	SQUIBB MFG. INC.	A	A02	1		09500	N
283	00661SQBBMSTATE	SQUIBB MFG. INC.	A	A03	1		09000	N
283	00661SQBBMSTATE	SQUIBB MFG. INC.	W	F01	4		10000	N
283	00661SQBBMSTATE	SQUIBB MFG. INC.	L	F01	1		10000	N
283	00661SQBBMSTATE	SQUIBB MFG. INC.	W	B11	3		09800	N
283	00671KYPHRPRIDC	SCHERING-PLOUGH PRODUCTS INC. KEY PHARMA	A	P12	1	Y	00000	Y
283	00671KYPHRPRIDC	SCHERING-PLOUGH PRODUCTS INC. KEY PHARMA	A	P21		Y	09800	Y
283	00671KYPHRPRIDC	SCHERING-PLOUGH PRODUCTS INC. KEY PHARMA	W	P42		Y	09800	Y
283	00671KYPHRPRIDC	SCHERING-PLOUGH PRODUCTS INC. KEY PHARMA	W	P19		Y	09800	Y
283	00671KYPHRPRIDC	SCHERING-PLOUGH PRODUCTS INC. KEY PHARMA	L	R13		Y	09800	Y
283	00671KYPHRPRIDC	SCHERING-PLOUGH PRODUCTS INC. KEY PHARMA	L	R11		Y	09900	Y
283	00671KYPHRPRIDC	SCHERING-PLOUGH PRODUCTS INC. KEY PHARMA	L	P01	2	Y	00000	N
283	00671KYPHRPRIDC	SCHERING-PLOUGH PRODUCTS INC. KEY PHARMA	L	B11	4	Y	09500	Y
283	00671KYPHRPRIDC	SCHERING-PLOUGH PRODUCTS INC. KEY PHARMA	L	P12		Y	00000	Y
283	00701RCHPRSTATE	ROCHE PRODUCTS INC.	A	P21	1		06000	N
283	00701RCHPRSTATE	ROCHE PRODUCTS INC.	W	P01	3	Y	00000	

Treatments for Methylene Chloride (CAS 000075092)

SIC	TRIS ID	FACILITY NAME	WC	WTC	IC	STI	EFFIC	OD
283	00701RCHPRSTATE	ROCHE PRODUCTS INC.	W	P11		Y	00000	
283	00701RCHPRSTATE	ROCHE PRODUCTS INC.	W	B11		Y	00000	
283	00701RCHPRSTATE	ROCHE PRODUCTS INC.	W	P13		Y	00000	
283	00701RCHPRSTATE	ROCHE PRODUCTS INC.	W	P11		Y	09100	Y
283	00701SCHRNROAD6	SCHERING INDUSTRIAL DEVELOPMENT CORP.		NA			00000	
283	00732BLCHMRDNO1	BILCHEM LTD.	A	A02	1		09000	N
283	00732BLCHMRDNO1	BILCHEM LTD.	W	C11	3		00000	N
283	06340PFZRNEASTE	PFIZER INC. GROTON SITE	G	A02	1		08500	N
283	06340PFZRNEASTE	PFIZER INC. GROTON SITE	G	A03	1		09200	N
283	06340PFZRNEASTE	PFIZER INC. GROTON SITE	W	R13	1		09900	N
283	06340PFZRNEASTE	PFIZER INC. GROTON SITE	G	A03	1		09200	N
283	06340PFZRNEASTE	PFIZER INC. GROTON SITE	W	R13	1		09900	N
283	06810DNBRY131WE	DANBURY PHARMACAL INC.		NA			00000	
283	07072RSYNCFOOTO	ARSYNCO INC.	W	P15	3	Y	00000	Y
283	07072RSYNCFOOTO	ARSYNCO INC.	W	C11		Y	00000	Y
283	07072RSYNCFOOTO	ARSYNCO INC.	W	P11		Y	00000	Y
283	07072RSYNCFOOTO	ARSYNCO INC.	W	P41			09985	Y
283	07080NTRLB650SO	NUTRO LABORATORIES INC.		NA			00000	
283	07083SCHRN1011M	SCHERING CORP.		NA			00000	
283	07083SCHRN1011M	SCHERING CORP.		NA			00000	
283	07110HFFMN340KI	HOFFMANN-LA ROCHE INC.	W	P01	2		00000	N
283	07110HFFMN340KI	HOFFMANN-LA ROCHE INC.	L	P09	1		00000	N
283	07110HFFMN340KI	HOFFMANN-LA ROCHE INC.	A	A02	1		08200	N
283	07110HFFMN340KI	HOFFMANN-LA ROCHE INC.	A	A02	1		07000	N
283	07110HFFMN340KI	HOFFMANN-LA ROCHE INC.	A	A02	1		09000	N
283	07110HFFMN340KI	HOFFMANN-LA ROCHE INC.	A	A02	1		07200	N
283	07110HFFMN340KI	HOFFMANN-LA ROCHE INC.	A	A02	1		09400	N
283	07110HFFMN340KI	HOFFMANN-LA ROCHE INC.	A	A07	1		00000	N
283	07110HFFMN340KI	HOFFMANN-LA ROCHE INC.	A	A02	1		08300	N
283	07110HFFMN340KI	HOFFMANN-LA ROCHE INC.	A	A02	1		09200	N
283	07110HFFMN340KI	HOFFMANN-LA ROCHE INC.	A	A02	1		06700	N
283	07110HFFMN340KI	HOFFMANN-LA ROCHE INC.	A	A02	1		00000	N
283	07110HFFMN340KI	HOFFMANN-LA ROCHE INC.	A	A07	1		09900	N
283	07110HFFMN340KI	HOFFMANN-LA ROCHE INC.	A	A03	1		09000	N
283	07110HFFMN340KI	HOFFMANN-LA ROCHE INC.	A	A02	1		04000	N
283	07110HFFMN340KI	HOFFMANN-LA ROCHE INC.		NA	NA		NA	N
283	07424MDPHR101EM	AMIDE PHARMACEUTICAL INC.		NA			00000	
283	07463BCRFT12IND	BIOCRAFT LABORATORIES INC.	A	A03	1		09500	N
283	07901CBGGY556MO	CIBA-GEIGY CORP. PHARMACEUTICALS DIV.	A	A02	1		09500	Y
283	07901CBGGY556MO	CIBA-GEIGY CORP. PHARMACEUTICALS DIV.	A	A04	1		09700	Y
283	07901CBGGY556MO	CIBA-GEIGY CORP. PHARMACEUTICALS DIV.	A	A07	1		09900	Y

Treatments for Methylene Chloride (CAS 000075092)

SIC	TRIS ID	FACILITY NAME	WC	WTC	IC	STI	EFFIC	OD
283	07901CBGGY556MO	CIBA-GEIGY CORP. PHARMACEUTICALS DIV.	W	P01	3		10000	Y
283	07901CBGGY556MO	CIBA-GEIGY CORP. PHARMACEUTICALS DIV.	L	R19	1		01800	Y
283	07936SDMKL17WES	SIDMAK LABORATORIES INC.		NA			00000	
283	07936SNDZP59ROU	SANDOZ PHARMACEUTICALS CORP.		NA			00000	
283	08807MRCNCEASTM	AMERICAN CYANAMID CO. LEDERLE LABORATORI	A	A07	1		09000	N
283	08807MRCNCEASTM	AMERICAN CYANAMID CO. LEDERLE LABORATORI	W	C11	3		00000	N
283	08807MRCNCEASTM	AMERICAN CYANAMID CO. LEDERLE LABORATORI	A	A07	1		09000	N
283	08807MRCNCEASTM	AMERICAN CYANAMID CO. LEDERLE LABORATORI	W	C11	3		00000	N
283	08807MRCNCEASTM	AMERICAN CYANAMID CO. LEDERLE LABORATORI		NA	NA		NA	N
283	08854BCHML101PO	SMITHKLINE BEECHAM PHARMACEUTICALS		NA			00000	
283	08854BCHML101PO	SMITHKLINE BEECHAM PHARMACEUTICALS		NA			00000	
283	08876HCHSTRTE20	HOECHST CELANESE CORP. LIFE SCIENCES		NA	NA		NA	N
283	08902RSQBBONESQ	E.R. SQUIBB & SONS	A	A02	1		09500	N
283	08902RSQBBONESQ	E.R. SQUIBB & SONS	A	A02	1		09900	N
283	08902RSQBBONESQ	E.R. SQUIBB & SONS	A	A04	1		09500	Y
283	08902RSQBBONESQ	E.R. SQUIBB & SONS	A	A02	1		09500	N
283	10901CBGGYOLDMI	CIBA-GEIGY CORP. PHARMACEUTICALS DIV.		NA			00000	
283	10920PRPHR75BRE	PAR PHARMACEUTICAL INC.		NA			00000	
283	10965LDRLLNORTH	LEDERLE LABORATORIES	W	B11	3		04700	Y
283	11590TSHCN125ST	TISHCON CORP.						
283	12979YRSTL64MAP	AYERST LABORATORIES INC.	A	A02	1		09400	N
283	13221BRSTLTHOMP	BRISTOL-MYERS SQUIBB CO. INDUSTRIAL DIV.	A	A02	2		09850	N
283	13221BRSTLTHOMP	BRISTOL-MYERS SQUIBB CO. INDUSTRIAL DIV.	A	A02	2		06300	N
283	13221BRSTLTHOMP	BRISTOL-MYERS SQUIBB CO. INDUSTRIAL DIV.	A	A03	2		00000	N
283	13221BRSTLTHOMP	BRISTOL-MYERS SQUIBB CO. INDUSTRIAL DIV.	A	A02	2		09900	N
283	13221BRSTLTHOMP	BRISTOL-MYERS SQUIBB CO. INDUSTRIAL DIV.	W	C11	3		00000	N
283	13221BRSTLTHOMP	BRISTOL-MYERS SQUIBB CO. INDUSTRIAL DIV.	A	A03	2		08750	N
283	13221BRSTLTHOMP	BRISTOL-MYERS SQUIBB CO. INDUSTRIAL DIV.	A	A02	2		09900	N
283	13221BRSTLTHOMP	BRISTOL-MYERS SQUIBB CO. INDUSTRIAL DIV.	A	A02	2		09800	N
283	13221BRSTLTHOMP	BRISTOL-MYERS SQUIBB CO. INDUSTRIAL DIV.	A	A02	2		08000	N
283	13221BRSTLTHOMP	BRISTOL-MYERS SQUIBB CO. INDUSTRIAL DIV.	A	A04	2		09500	N
283	13221BRSTLTHOMP	BRISTOL-MYERS SQUIBB CO. INDUSTRIAL DIV.	W	C11	2		00000	N
283	14623PNNWL755JE	FISONS CORP.	A	A04	2		09600	Y
283	14623PNNWL755JE	FISONS CORP.	A	P42	1		10000	Y
283	14623PNNWL755JE	FISONS CORP.	W	P12	1		08500	Y
283	15147PNNXPEASTE	PENNEX PRODUCTS CO. INC.					NA	
283	17868MRCKC100AV	MERCK & CO. INC.	W	P01	3	Y	NA	
283	17868MRCKC100AV	MERCK & CO. INC.	W	C11		Y	NA	
283	17868MRCKC100AV	MERCK & CO. INC.	W	P11		Y	NA	
283	17868MRCKC100AV	MERCK & CO. INC.	W	B11		Y	NA	
283	17868MRCKC100AV	MERCK & CO. INC.	W	P11		Y	09997	Y

Treatments for Methylene Chloride (CAS 000075092)

SIC	TRIS ID	FACILITY NAME	WC	WTC	IC	STI	EFFIC	OD
283	17868MRCKC100AV	MERCK & CO. INC.	A	A02	2		08300	N
283	17868MRCKC100AV	MERCK & CO. INC.	A	A02	2		09100	N
283	19034RRRPH500VI	RHONE-POULENC RORER PHARMACEUT ICAL INC.		NA			00000	
283	19130SMTHK1500S	SMITHKLINE BEECHAM PHARMACEUTICALS	A	F71	2	Y	00000	
283	19130SMTHK1500S	SMITHKLINE BEECHAM PHARMACEUTICALS	A	A03		Y	08690	Y
283	19130SMTHK1500S	SMITHKLINE BEECHAM PHARMACEUTICALS		NA			00000	
283	19382FRMNT510EU	FERMTEC PRODUCTS INC.	W	B11	3		10000	N
283	19382FRMNT510EU	FERMTEC PRODUCTS INC.	W	B11	3		10000	N
283	19382WYTHY611EN	WYETH-AYERST LABORATORIES INC.	W	P01	3	Y	00000	
283	19382WYTHY611EN	WYETH-AYERST LABORATORIES INC.	W	B11		Y	00000	
283	19382WYTHY611EN	WYETH-AYERST LABORATORIES INC.	W	P14		Y	09500	N
283	19382WYTHY611EN	WYETH-AYERST LABORATORIES INC.	W	P01	3	Y	00000	
283	19382WYTHY611EN	WYETH-AYERST LABORATORIES INC.	W	B11		Y	00000	
283	19382WYTHY611EN	WYETH-AYERST LABORATORIES INC.	W	P14		Y	09000	N
283	19428SMTHK900RI	SMITHKLINE BEECHAM PHARMACEUTI CALS	W	F01	2		00100	Y
283	19428SMTHK900RI	SMITHKLINE BEECHAM PHARMACEUTI CALS	A	A02	1		00096	N
283	19428SMTHK900RI	SMITHKLINE BEECHAM PHARMACEUTI CALS	A	A03	2		00005	N
283	19428SMTHK900RI	SMITHKLINE BEECHAM PHARMACEUTI CALS		NA	NA		00000	N
283	19801NRMCF5000L	NORAMCO OF DELAWARE INC.		NA			00000	
283	21225KNSCL6118R	KANASCO LTD.		NA			NA	
283	21225KNSCL6118R	KANASCO LTD.		NA			00000	
283	23805LLBRT2999F	LEE LABORATORIES INC. & INFRACORP LTD.	W	P15	4	Y	NA	
283	23805LLBRT2999F	LEE LABORATORIES INC. & INFRACORP LTD.	W	C11		Y	00000	N
283	23805LLBRT2999F	LEE LABORATORIES INC. & INFRACORP LTD.	A	A03	3		00000	N
283	27597GLXNC1011N	GLAXO INC. TECHNICAL OPERATIONS DIV.	A	A02	1	Y	00000	
283	27597GLXNC1011N	GLAXO INC. TECHNICAL OPERATIONS DIV.		A04		Y	00100	N
283	27597GLXNC1011N	GLAXO INC. TECHNICAL OPERATIONS DIV.					00000	
283	27597GLXNC1011N	GLAXO INC. TECHNICAL OPERATIONS DIV.					00000	
283	27835BRRGHINTER	BURROUGHS WELLCOME CO.	W	C11	2		03200	N
283	27835BRRGHINTER	BURROUGHS WELLCOME CO.	A	A02	1		10000	N
283	27835BRRGHINTER	BURROUGHS WELLCOME CO.	L	R11	1		09000	N
283	27835BRRGHINTER	BURROUGHS WELLCOME CO.	L	F01	1	Y	NA	
283	27835BRRGHINTER	BURROUGHS WELLCOME CO.	A	A03		Y	10000	Y
283	31708MRCKC3517R	MERCK & CO. INC. FLINT RIVER PLANT	W	P01	2	Y	NA	
283	31708MRCKC3517R	MERCK & CO. INC. FLINT RIVER PLANT	W	C11		Y	NA	
283	31708MRCKC3517R	MERCK & CO. INC. FLINT RIVER PLANT	W	B11		Y	NA	
283	31708MRCKC3517R	MERCK & CO. INC. FLINT RIVER PLANT	W	P11		Y	NA	
283	31708MRCKC3517R	MERCK & CO. INC. FLINT RIVER PLANT	S	P13		Y	09500	N
283	37620BCHMLINDUS	BEECHAM LABORATORIES	A	A03	1		00140	N
283	37620BCHMLINDUS	BEECHAM LABORATORIES	W	A03	1		09860	N
283	45215MRRLL2110E	MERRELL DOW PHARMACEUTICALS INC.	A	A03	2		00000	N

Treatments for Methylene Chloride (CAS 000075092)

SIC	TRIS ID	FACILITY NAME	WC	WTC	IC	STI	EFFIC	OD
283	45215MRRLL2110E	MERRELL DOW PHARMACEUTICALS INC.		NA			00000	
283	46285LLLLY1555K	ELI LILLY & CO. LILLY INDUSTRIAL CENTER	A	A02	1		07500	N
283	46285LLLLY1555K	ELI LILLY & CO. LILLY INDUSTRIAL CENTER		NA	NA		NA	N
283	46285LLLLYLILLY	ELI LILLY & CO. LILLY CORPORATE CENTER	A	A07	3		10000	N
283	46285LLLLYLILLY	ELI LILLY & CO. LILLY CORPORATE CENTER	A	A03	3		05000	N
283	46285LLLLYLILLY	ELI LILLY & CO. LILLY CORPORATE CENTER	L	R12	1	Y	NA	N
283	46285LLLLYLILLY	ELI LILLY & CO. LILLY CORPORATE CENTER	A	A02	NA	Y	10000	N
283	46285LLLLYLILLY	ELI LILLY & CO. LILLY CORPORATE CENTER		NA	NA		NA	N
283	47721BRSTL2404P	MEAD JOHNSON & CO. EVANSVILLE PLANT	A	A03	2		09000	N
283	47721BRSTL2404P	MEAD JOHNSON & CO. EVANSVILLE PLANT		NA			00000	
283	47842LLLLYSTATE	ELI LILLY & CO. CLINTON LABORATORIES	W	F01	1		10000	N
283	47842LLLLYSTATE	ELI LILLY & CO. CLINTON LABORATORIES	L	F01	1		10000	N
283	47842LLLLYSTATE	ELI LILLY & CO. CLINTON LABORATORIES	A	F71	1		10000	N
283	47842LLLLYSTATE	ELI LILLY & CO. CLINTON LABORATORIES	A	A02	1		08500	N
283	47842LLLLYSTATE	ELI LILLY & CO. CLINTON LABORATORIES	L	R11	1		07300	Y
283	47842LLLLYSTATE	ELI LILLY & CO. CLINTON LABORATORIES		NA	NA		NA	N
283	47905LLLLYLILLY	ELI LILLY & CO. TIPPECANOE LABORATORIES	A	A02	1		08500	N
283	47905LLLLYLILLY	ELI LILLY & CO. TIPPECANOE LABORATORIES	A	A03	1		01000	N
283	47905LLLLYLILLY	ELI LILLY & CO. TIPPECANOE LABORATORIES	W	B11	3		00000	N
283	47905LLLLYLILLY	ELI LILLY & CO. TIPPECANOE LABORATORIES	W	C11	3		00000	N
283	47905LLLLYLILLY	ELI LILLY & CO. TIPPECANOE LABORATORIES	L	F01	1		09900	N
283	47905LLLLYLILLY	ELI LILLY & CO. TIPPECANOE LABORATORIES	L	P01	1		00000	N
283	47905LLLLYLILLY	ELI LILLY & CO. TIPPECANOE LABORATORIES	W	P01	3		00000	N
283	47905LLLLYLILLY	ELI LILLY & CO. TIPPECANOE LABORATORIES	W	P11	3		00500	N
283	47905LLLLYLILLY	ELI LILLY & CO. TIPPECANOE LABORATORIES	S	P13	3		01800	N
283	47905LLLLYLILLY	ELI LILLY & CO. TIPPECANOE LABORATORIES	W	F01	2		09900	N
283	47905LLLLYLILLY	ELI LILLY & CO. TIPPECANOE LABORATORIES	W	P41	3		09000	N
283	47905LLLLYLILLY	ELI LILLY & CO. TIPPECANOE LABORATORIES	W	P42	1		09000	N
283	47905LLLLYLILLY	ELI LILLY & CO. TIPPECANOE LABORATORIES	L	R13	1		09000	N
283	47905LLLLYLILLY	ELI LILLY & CO. TIPPECANOE LABORATORIES	A	P21	1		08000	N
283	47905LLLLYLILLY	ELI LILLY & CO. TIPPECANOE LABORATORIES	L	C11	1		00000	N
283	47905LLLLYLILLY	ELI LILLY & CO. TIPPECANOE LABORATORIES		NA	NA		NA	N
283	49001THPJH7171P	UPJOHN CO. PRODUCTION FACILITY	L	R13	1		08500	Y
283	49001THPJH7171P	UPJOHN CO. PRODUCTION FACILITY	L	R11	1		08800	Y
283	49001THPJH7171P	UPJOHN CO. PRODUCTION FACILITY	L	F19	2		09999	N
283	49001THPJH7171P	UPJOHN CO. PRODUCTION FACILITY	A	A02	1		08000	N
283	49001THPJH7171P	UPJOHN CO. PRODUCTION FACILITY	A	A02	1	Y	NA	N
283	49001THPJH7171P	UPJOHN CO. PRODUCTION FACILITY	A	A03	NA	Y	09900	N
283	49001THPJH7171P	UPJOHN CO. PRODUCTION FACILITY		NA	NA		NA	N
283	49001THPJH7171P	UPJOHN CO. PRODUCTION FACILITY	A	A02	1		08000	N
283	49001THPJH7171P	UPJOHN CO. PRODUCTION FACILITY		NA	NA		NA	N

Treatments for Methylene Chloride (CAS 000075092)

SIC	TRIS ID	FACILITY NAME	WC	WTC	IC	STI	EFFIC	OD
283	49424PRKDV188HO	PARKE-DAVIS DIV. OF WARNER-LAMBERT CO.	W	P01	3		00000	N
283	49424PRKDV188HO	PARKE-DAVIS DIV. OF WARNER-LAMBERT CO.	S	P12	3		00000	N
283	49424PRKDV188HO	PARKE-DAVIS DIV. OF WARNER-LAMBERT CO.	A	F71	1		09999	Y
283	49424PRKDV188HO	PARKE-DAVIS DIV. OF WARNER-LAMBERT CO.	L	R11	1		09500	N
283	55447PSHRS14905	UPSHER-SMITH LABORATORIES INC.		NA			00000	
283	56623RDRWL210MA	REID-ROWELL INC.		NA			00000	
283	60064BBTTL1400N	ABBOTT LABORATORIES	W	B11	2		09930	Y
283	60064BBTTL1400N	ABBOTT LABORATORIES	A	A02	1		09500	N
283	60077GDSRL4901S	G. D. SEARLE & CO.	A	A02	1		07000	N
283	60077GDSRL4901S	G. D. SEARLE & CO.		NA	NA		NA	N
283	60915RMRPHRT50A	ARMOUR PHARMACEUTICAL CO.		NA			00000	
283	63141KVPHR2303S	KV PHARMACEUTICAL CO.	A	A07	3		00000	Y
283	64137MRNLBMARIO	MARION MERRELL DOW INC. MARION PARK		NA			00000	
283	65807SYNTX2460W	SYNTEX AGRIBUSINESS INC.	W	P41	3		08000	N
283	65807SYNTX2460W	SYNTEX AGRIBUSINESS INC.	A	A03	2		00300	N
283	68521NRDNL601WE	SMITHKLINE BEECHAM ANIMAL HEALTH		NA			00000	
283	80020CRDLB2555W	GENEVA PHARMACEUTICALS INC.		NA			00000	
283	80301HSRCH4750N	HAUSER CHEMICAL RESEARCH INC.	L	R11	1		09600	Y
283	80301SYNTX2075N	SYNTEX CHEMICALS INC.	A	A03	2	Y	NA	N
283	80301SYNTX2075N	SYNTEX CHEMICALS INC.	A	A04	NA	Y	00500	N
283	80301SYNTX2075N	SYNTEX CHEMICALS INC.	A	A04	2		00000	N
283	80301SYNTX2075N	SYNTEX CHEMICALS INC.	W	B11	2		09600	N
283	80301SYNTX2075N	SYNTEX CHEMICALS INC.		NA	NA		NA	N
283	85260NTRLL14810	NATURALLY VITAMIN SUPPLEMENTS INC.		NA			00000	
283	90505BCHMN3132K	BACHEM INC.		NA			00000	
283	91324MRKR 19901	3M RIKER		NA			00000	
283	92121MLTPL3550G	MULTIPLE PEPTIDE SYSTEMS L.P.	L	R11	1		08300	N
283	92713NBLCN17802	ANABOLIC INC.	A	P41	2		00130	N
283	94303LZCRP2575H	ALZA CORP.	A	F71	1	Y	NA	
283	94303LZCRP2575H	ALZA CORP.	A	A03		Y	10000	Y
283	95688LZCRP700EU	ALZA CORP.	A	A02	1		09300	Y

---------- SIC=283 ----------

SIC	TRIS ID	FACILITY NAME	LOCATION	PARENT COMPANY
283	85260NTRLL14810	NATURALLY VITAMIN SUPPLEMENTS INC.	SCOTTSDALE, AZ	MARLYN CO.
283	37620BCHMLINDUS	BEECHAM LABORATORIES	BRISTOL, TN	SMITHKLINE BEECHAM PHARMACEUTI
283	14623PNNWL755JE	FISONS CORP.	ROCHESTER, NY	FISONS CORP.
283	15147PNNXPEASTE	PENNEX PRODUCTS CO. INC.	VERONA, PA	NV MEDICOPHARMA
283	06810DNBRY131WE	DANBURY PHARMACAL INC.	DANBURY, CT	HENRY SCHEIN INC.
283	80020CRDLB2555W	GENEVA PHARMACEUTICALS INC.	BROOMFIELD, CO	CIBA GEIGY CORP.
283	19034RRRPH500VI	RHONE-POULENC RORER PHARMACEUT ICAL INC.	FORT WASHINGTON, PA	RHONE-POULENC RORER INC.
283	31708MRCKC3517R	MERCK & CO. INC. FLINT RIVER PLANT	ALBANY, GA	MERCK & CO. INC.
283	23805LLBRT2999F	LEE LABORATORIES INC. & INFRACORP LTD.	PETERSBURG, VA	BOEHRINGER INGELHEIM & A. H. R
283	07080NTRLB650SO	NUTRO LABORATORIES INC.	SOUTH PLAINFIELD, NJ	NA
283	07936SNDZP59ROU	SANDOZ PHARMACEUTICALS CORP.	EAST HANOVER, NJ	SANDOZ CORP.
283	00617MRCKSSTATE	MERCK SHARP & DOHME QUIMICA DE P.R.	BARCELONETA, PR	MERCK & CO. INC.
283	55447PSHRS14905	UPSHER-SMITH LABORATORIES INC.	MINNEAPOLIS, MN	NA
283	12979YRSTL64MAP	AYERST LABORATORIES INC.	ROUSES POINT, NY	AMERICAN HOME PRODUCTS CORP.

ONSITE	FUGITIVE EMISSIONS	FC	STACK EMISSIONS	SC	TOTAL EMISSIONS
02	0	O	100	O	100
04	0		184028	M	184028
04	250	O	2016	O	2266
	1148	C	140000	C	141148
03	250	O	162000	C	162250
03	0		14412	C	14412
03	0		31367	C	31367
06	250000	O	1200000	O	1450000
04	51	O	7900	O	7951
03	0		30000	C	30000
04	93	O	2250	O	2343
04	650	M	270	O	920
03	0		120000	C	120000
04	250	O	16000	E	16250

SIC=283
(continued)

SIC	TRIS ID	FACILITY NAME
283	17868MRCKC100AV	MERCK & CO. INC.
283	06340PFZRNEASTE	PFIZER INC. GROTON SITE
283	07936SDMKL17WES	SIDMAK LABORATORIES INC.
283	10920PRPHR75BRE	PAR PHARMACEUTICAL INC.
283	64137MRNLBMARIO	MARION MERRELL DOW INC. MARION PARK
283	90505BCHMN3132K	BACHEM INC.
283	68521NRDNL601WE	SMITHKLINE BEECHAM ANIMAL HEALTH
283	60915RMRPHRT50A	ARMOUR PHARMACEUTICAL CO.
283	00701SCHRNROAD6	SCHERING INDUSTRIAL DEVELOPMENT CORP.
283	07424MDPHR101EM	AMIDE PHARMACEUTICAL INC.
283	65807SYNTX2460W	SYNTEX AGRIBUSINESS INC.
283	00617THPJHHIGHW	UPJOHN MFG. CO.
283	19382FRMNT510EU	FERMTEC PRODUCTS INC.

LOCATION	PARENT COMPANY
RIVERSIDE, PA	MERCK & CO. INC.
GROTON, CT	PFIZER INC.
EAST HANOVER, NJ	NA
CONGERS, NY	NA
KANSAS CITY, MO	DOW CHEMICAL CO.
TORRANCE, CA	NA
LINCOLN, NE	SMITHKLINE BEECHAM CORP.
BRADLEY, IL	RHONE-POULENC RORER INC.
MANATI, PR	SCHERING INDUSTRIAL DEVELOPMEN
LITTLE FALLS, NJ	NA
SPRINGFIELD, MO	SYNTEX USA INC.
BARCELONETA, PR	UPJOHN CO.
WEST CHESTER, PA	BURNS PHILP & CO. LTD.

ONSITE	FUGITIVE EMISSIONS	FC	STACK EMISSIONS	SC	TOTAL EMISSIONS
05	200000	O	110000	O	310000
05	17000	M	8500	O	25500
03	4395	C	171398	C	175793
03	200	O	1300	C	1500
04	750	E	58000	C	58750
03	5	O	250	O	255
04	5200	O	1080500	C	1085700
04	0		9600	O	9600
04	250	O	27558	C	27808
03	2192	O	19730	O	21922
05	31243	E	821777	C	853020
06	4995	E	584777	E	589772
04	0		250	E	250

SIC=283
(continued)

SIC	TRIS ID	FACILITY NAME
283	10901CBGGYOLDMI	CIBA-GEIGY CORP. PHARMACEUTICALS DIV.
283	27835BRRGHINTER	BURROUGHS WELLCOME CO.
283	07083SCHRN1011M	SCHERING CORP.
283	00617PFZRPHIGHW	PFIZER PHARMACEUTICALS INC.
283	80301HSRCH4750N	HAUSER CHEMICAL RESEARCH INC.
283	27597GLXNC1011N	GLAXO INC. TECHNICAL OPERATIONS DIV.
283	94303LZCRP2575H	ALZA CORP.
283	45215MRRLL2110E	MERRELL DOW PHARMACEUTICALS INC.
283	80301SYNTX2075N	SYNTEX CHEMICALS INC.
283	19382WYTHY611EN	WYETH-AYERST LABORATORIES INC.
283	07463BCRFT12IND	BIOCRAFT LABORATORIES INC.
283	08902RSQBBONESQ	E.R. SQUIBB & SONS
283	47721BRSTL2404P	MEAD JOHNSON & CO. EVANSVILLE PLANT

LOCATION	PARENT COMPANY
SUFFERN, NY	CIBA-GEIGY CORP. USA
GREENVILLE, NC	WELLCOME FOUNDATION LTD.
UNION, NJ	SCHERING-PLOUGH CORP.
BARCELONETA, PR	PFIZER INC.
BOULDER, CO	NA
ZEBULON, NC	GLAXO ENTERPRISES INC.
PALO ALTO, CA	NA
CINCINNATI, OH	MARION MERRELL DOW INC.
BOULDER, CO	SYNTEX (USA) INC.
WEST CHESTER, PA	AMERICAN HOME PRODUCTS
WALDWICK, NJ	BIOCRAFT LABORATORIES INC.
NORTH BRUNSWICK, NJ	BRISTOL-MYERS SQUIBB CO.
EVANSVILLE, IN	BRISTOL-MYERS SQUIBB CO.

ONSITE	FUGITIVE EMISSIONS	FC	STACK EMISSIONS	SC	TOTAL EMISSIONS
03	2974	O	14978	C	17952
04	13000	C	433000	C	446000
04	0	O	76263	O	76263
05	24000	O	37100	O	61100
04	61400	CM	0		61400
04	2290	O	155334	C	157624
04	250	O	3434	O	3684
04	400	O	25000	C	25400
05	250	O	61300	O	61550
05	4800	O	28400	O	33200
04	2514	O	6694	O	9208
04	4100	O	4100	O	8200
04	250	O	51300	C	51550

SIC=283
(continued)

SIC	TRIS ID	FACILITY NAME	LOCATION	PARENT COMPANY	ONSITE	FUGITIVE EMISSIONS	FC	STACK EMISSIONS	SC	TOTAL EMISSIONS
283	00617BBTTCROADN	ABBOTT CHEMICALS INC.	BARCELONETA, PR	ABBOTT LABORATORIES INC.	05	784762	C	1741170	C	2525932
283	49424PRKDV188HO	PARKE-DAVIS DIV. OF WARNER-LAMBERT CO.	HOLLAND, MI	WARNER LAMBERT CO.	04	1100	O	5	O	1105
283	07901CBGGY556MO	CIBA-GEIGY CORP. PHARMACEUTICALS DIV.	SUMMIT, NJ	CIBA-GEIGY CORP.	05	0	C	13400	C	13400
283	13221BRSTLTHOMP	BRISTOL-MYERS SQUIBB CO. INDUSTRIAL DIV.	SYRACUSE, NY	BRISTOL-MYERS SQUIBB CO.	05	190000	C	110000	E	300000
283	10965LDRLLNORTH	LEDERLE LABORATORIES	PEARL RIVER, NY	AMERICAN CYANAMID CO.	04	0	O	3300	O	3300
283	60064BBTTL1400N	ABBOTT LABORATORIES	NORTH CHICAGO, IL	ABBOTT LABORATORIES INC.	05	7700	O	117550	O	125250
283	00701RCHPRSTATE	ROCHE PRODUCTS INC.	MANATI, PR	HOFFMANN-LA ROCHE LTD.	04	4914	O	13608	O	18522
283	21225KNSCL6118R	KANASCO LTD.	BALTIMORE, MD		05	1500	O	1400	O	2900
283	56623RDRWL210MA	REID-ROWELL INC.	BAUDETTE, MN	SOLVAY PHARMACEUTICAL INC.	04	0		100117	C	100117
283	19801NRMCF5000L	NORAMCO OF DELAWARE INC.	WILMINGTON, DE	JOHNSON & JOHNSON	05	11362	E	19815	E	31177
283	00671KYPHRPRIDC	SCHERING-PLOUGH PRODUCTS INC. KEY PHARMA	LAS PIEDRAS, PR	SCHERING CORP.	05	41000	E	54800	O	95800
283	00661SQBBMSTATE	SQUIBB MFG. INC.	HUMACAO, PR	BRISTOL-MYERS SQUIBB CO.	05	95640	O	93260	O	188900
283	07072RSYNCFOOTO	ARSYNCO INC.	CARLSTADT, NJ	ACETO CHEMICAL CO.	04	108778	O	16821	O	125599

SIC=283
(continued)

SIC	TRIS ID	FACILITY NAME	LOCATION	PARENT COMPANY	ONSITE	FUGITIVE EMISSIONS	FC	STACK EMISSIONS	SC	TOTAL EMISSIONS
283	08807MRCNCEASTM	AMERICAN CYANAMID CO. LEDERLE LABORATORI	BRIDGEWATER, NJ	AMERICAN CYANAMID CO.	04	343	O	27345	O	27688
283	11590TSHCN125ST	TISHCON CORP.	WESTBURY, NY	TISHCON CORP.	03	0		190971	C	190971
283	91324MRKR 19901	3M RIKER	NORTHRIDGE, CA	3M CO.	03	1560	C	15600	C	17160
283	47905LLLLYLILLY	ELI LILLY & CO. TIPPECANOE LABORATORIES	SHADELAND, IN	ELI LILLY &. CO.	06	180000	O	530000	O	710000
283	19428SMTHK900RI	SMITHKLINE BEECHAM PHARMACEUTI CALS	CONSHOHOCKEN, PA	SMITHKLINE BEECHAM PHARMACEUTI	04	250	E	1700	O	1950
283	60077GDSRL4901S	G. D. SEARLE & CO.	SKOKIE, IL	MONSANTO CO.	04	3200	O	1200	O	4400
283	00628LLLLY65THI	ELI LILLY INDUSTRIES INC.	CAROLINA, PR	ELI LILLY &. CO.	03	18000	O	17000	C	35000
283	08807MRCNCEASTM	AMERICAN CYANAMID CO. LEDERLE LABORATORI	BRIDGEWATER, NJ	AMERICAN CYANAMID CO.	04	597	O	28905	O	29502
283	07110HFFMN340KI	HOFFMANN-LA ROCHE INC.	NUTLEY, NJ	ROCHE HOLDINGS INC.	05	12369	O	6372	O	18741
283	49001THPJH7171P	UPJOHN CO. PRODUCTION FACILITY	PORTAGE, MI	UPJOHN CO.	06	662600	E	2302300	O	2964900
283	46285LLLLYLILLY	ELI LILLY & CO. LILLY CORPORATE CENTER	INDIANAPOLIS, IN	ELI LILLY & CO.	03	6500	C	1200	C	7700
283	46285LLLLY1555K	ELI LILLY & CO. LILLY INDUSTRIAL CENTER	INDIANAPOLIS, IN	ELI LILLY & CO.	04	11000	O	200000	O	211000
283	47842LLLLYSTATE	ELI LILLY & CO. CLINTON LABORATORIES	CLINTON, IN	ELI LILLY & CO.	06	1600000	C	810000	O	2410000

SIC=283
(continued)

SIC	TRIS ID	FACILITY NAME
283	95688LZCRP700EU	ALZA CORP.
283	19130SMTHK1500S	SMITHKLINE BEECHAM PHARMACEUTICALS
283	08854BCHML101PO	SMITHKLINE BEECHAM PHARMACEUTICALS
283	92121MLTPL3550G	MULTIPLE PEPTIDE SYSTEMS L.P.
283	92713NBLCN17802	ANABOLIC INC.
283	63141KVPHR2303S	KV PHARMACEUTICAL CO.
283	00732BLCHMRDNO1	BILCHEM LTD.
283	08876HCHSTRTE20	HOECHST CELANESE CORP. LIFE SCIENCES

SIC

LOCATION	PARENT COMPANY
VACAVILLE, CA	NA
PHILADELPHIA, PA	SMITHKLINE BEECHAM CORP.
PISCATAWAY, NJ	SMITHKLINE BEECHMAN CORP.
SAN DIEGO, CA	NA
IRVINE, CA	NA
SAINT LOUIS, MO	NA
PONCE, PR	BOHERINGER INGELHEIM INC.
SOMERVILLE, NJ	HOECHST CELANESE CORP.

ONSITE	FUGITIVE EMISSIONS	FC	STACK EMISSIONS	SC	TOTAL EMISSIONS
05	3563	O	24693	O	28256
04	5	O	55000	C	55005
05	750	O	5100	E	5850
03	750	O	0		750
04	250	O	14900	M	15150
04	0		299733	C	299733
04	1410	O	27229	O	28639
03	250	C	0	NA	250
	4385303		12545380		16930683

Key:

SIC - Standard Industrial Classification

TRIS - Toxic Chemical Release Inventory System

CAS - Chemical Abstract Service Registry Number

WC - General Wastestream type; where

A = Gaseous (gases, vapors, airborne particulates),
W = Wastewater (aqueous waste),
L = Liquid Waste (non-aqueous waste), and
S = Solid Waste (including sludges and slurries).

WTC - Waste Treatment Code; as indicated by the following.

Air Emissions Treatment

A01 Flare
A02 Condenser
A03 Scrubber
A04 Absorber
A05 Electrostatic Precipitator
A06 Mechanical Separation
A07 Other Air Emission Treatment

Biological Treatment

B11 Biological Treatment -- Aerobic
B21 Biological Treatment -- Anaerobic
B31 Biological Treatment -- Facultative
B99 Biological Treatment -- Other

Chemical Treatment

C01 Chemical Precipitation -- Lime or Sodium Hydroxide
C02 Chemical Precipitation -- Sulfide
C09 Chemical Precipitation -- Other
C11 Neutralization
C21 Chromium Reduction
C31 Complexed Metals Treatment (other than pH Adjustment)
C41 Cyanide Oxidation -- Alkaline Chlorination
C42 Cyanide Oxidation -- Electrochemical
C43 Cyanide Oxidation -- Other
C44 General Oxidation (including Disinfection) -- Chlorination
C45 General Oxidation (including Disinfection) -- Ozonation
C46 General Oxidation (including Disinfection) -- Other
C99 Other Chemical Treatment

Recovery/Reuse

R01 Reuse as Fuel -- Industrial Kiln
R02 Reuse as Fuel -- Industrial Furnace
R03 Reuse as Fuel -- Boiler
R04 Reuse as Fuel -- Fuel Blending
R09 Reuse as Fuel -- Other
R11 Solvents/Organics Recovery -- Batch Still Distillation
R12 Solvents/Organics Recovery -- Thin-Film Evaporation
R13 Solvents/Organics Recovery -- Fractionation
R14 Solvents/Organics Recovery -- Solvent Extraction
R19 Solvents/Organics Recovery -- Other
R21 Metals Recovery -- Electrolytic
R22 Metals Recovery -- Ion Exchange
R23 Metals Recovery -- Acid Leaching
R24 Metals Recovery -- Reverse Osmosis
R26 Metals Recovery -- Solvent Extraction
R29 Metals Recovery -- Other
R99 Other Reuse or Recovery

Solidification/Stabilization

G01 Cement Processes (including Silicates)
G09 Other Pozzolonic Processes (including Silicates)
G11 Asphaltic Processes
G21 Thermoplastic Techniques
G99 Other Solidification Processes

Incineration/Thermal Treatment

F01 Liquid Injection
F11 Rotary Kiln with Liquid Injection Unit
F19 Other Rotary Kiln
F31 Two Stage
F41 Fixed Hearth
F42 Multiple Hearth
F51 Fluidized Bed
F61 Infra-Red
F71 Fume/Vapor
F81 Pyrolytic Destructor
F82 Wet Air Oxidation
F83 Thermal Drying/Dewatering
F99 Other Incineration/Thermal Treatment

Physical Treatment

P01 Equalization
P09 Other Blending
P11 Settling/Clarification
P12 Filtration
P13 Sludge Dewatering (non-thermal)
P14 Air Flotation
P15 Oil Skimming
P16 Emulsion Breaking -- Thermal
P17 Emulsion Breaking -- Chemical
P18 Emulsion Breaking -- Other
P19 Other Liquid Phase Separation
P21 Adsorption -- Carbon
P22 Adsorption -- Ion Exchange (other than for recovery/reuse)
P23 Adsorption -- Resin
P29 Adsorption -- Other
P31 Reverse Osmosis (other than for recovery/reuse)
P41 Stripping -- Air
P42 Stripping -- Steam
P49 Stripping -- Other
P51 Acid Leaching (other than for recovery/reuse)
P61 Solvent Extraction (other than for recovery/reuse)
P99 Other Physical Treatment

IC = Influent Concentration; where

1 = Greater than 1%,
2 = 100 parts per million (ppm; 0.01%) to 1% (10,000 ppm),
3 = 1 ppm to 100 ppm,
4 = 1 part per billion (ppb) to 1 ppm, and
5 = Less than 1 ppb.

STI = Sequential Treatment; where

Y indicates that individual treatment steps are used in a series to treat the toxic chemical and there is no data on the efficiency of each step, however, an estimate of overall efficiency of the treatment sequence can be estimated.

EFFIC = Efficiency Estimate (i.e., refers to the percent destruction, degradation, conversion, or removal of the listed toxic chemical from the waste stream.

OD = Operating Data; where

Y = The treatment efficiency estimate is based on actual operating data,

N = The treatment efficiency is not based on actual operating data, and

NA = Not applicable.

Parent Company = the highest level company, located in the Unites States, that directly owns at least 50% of the voting stock.

On-Site = The maximum quantity of the chemical (e.g., in storage tanks, process vessels, on-site shipping containers) at your facility at any time during the calendar year. The following codes represent the weight range in pounds on-site.

WEIGHT RANGE IN POUNDS

Range Code	From...	To...
01	0	99
02	100	999
03	1,000	9,999
04	10,000	99,999
05	100,000	999,999
06	1,000,000	9,999,999
07	10,000,000	49,999,999
08	50,000,000	99,999,999
09	100,000,000	499,999,999
10	500,000,000	999,999,999
11	1 billion	more than 1 billion

Fugitive Emissions = all releases to the air that are not released through stacks, vents, ducts, pipes, or any other confined air stream (pounds/year).

FC = Basis of Fugitive Emissions Estimate.

SC = Basis of Stack Emissions Estimate.

FC and SC codes are as follows:

M - Estimate is based on monitoring data or measurements for the toxic chemical as released to the environment and/or off-site facility.

C - Estimate is based on mass balance calculations, such as calculation of the amount of the toxic chemical in streams entering and leaving process equipment.

E - Estimate is based on published emission factors, such as those relating release quantity to through-put or equipment type (e.g., air emission factors).

O - Estimate is based on other approaches such as engineering calculations (e.g., estimating volatilization using published mathematical formulas) or best engineering judgement. This would include applying an estimated removal efficiency to a waste stream, even if the composition of the stream before treatment was fully identified through monitoring data.

Source: Toxic Chemical Release Inventory Reporting Form R and Instructions, Revised 1990 Version, Section 313 of the Emergency Planning and Community Right-to-Know Act (Title III of the Superfund Amendments and Reauthorization Act of 1986). U.S. Environmental Protection Agency, Office of Toxic Substances, Washington, DC, EPA 560/4-91-007, January 1991.