Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

Title of Thesis : A computer program on the Economics of collection and sortation of recyclable materials in municipal solid waste.

Ara N. Kahyaoglu, Master of Science in Chemical Engineering, 1989

Thesis directed by Dr. Edward C. Roche, Jr.

5 May 89.
Date

The first version of the economic of recyclable materials in municipal solid waste analysis program was developed in interpretive BASIC by Rutgers the State University of New Jersey. Since BASIC is a line interpretative language several restrictions were encountered, the major one being the language size restriction.

The first objective was to obtain a truly portable program that would not be dependent on specific compilers and operating systems. The second objective was to generate a program that had the capability for expansion.

The C language was used over other high level languages such as PASCAL and FORTRAN because it had specific advantages over them. The language C has better file handling and ability to analyze character strings.

Comparison and analysis of each code (BASIC and C) was executed using the identical data sets. The same results were obtained in each case indicating the program conversion was successful.

It was concluded that the second version of this Economic Analysis Program is more suitable than the first one because of improved file manipulation and file checking.

A COMPUTER PROGRAM ON THE ECONOMICS OF COLLECTION AND SORTATION OF RECYCLABLE MATERIALS IN MUNICIPAL SOLID WASTES

By
Ara N. Kahyaoglu

Thesis submitted to the faculty of the Graduate School of the New Jersey Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Chemical Engineering
1989

APPROVAL SHEET

Title of Thesis: A computer program on the Economics of collection and sortation of recyclable materials in Municipal solid wastes.

Name of candidate: Ara Nigogos Kahyaoglu
Master of Science in Chemical Engineering,1989.

Thesis and Abstract approved: _____, 5 May 1989
Edward C.Roche, Jr. Date
Professor
Dept. of Chemical
Engineering, Chemistry,
and Environmental Science

_____ 5/5/89
Deran Hanesian Date
Professor
Dept. of Chemical
Engineering, Chemistry,
and Environmental Science

_____, 5/6/89
Irwin Hundert Date
Visiting Professor
Dept. of Chemical
Engineering, Chemistry,
and Environmental Science.

VITA

Name : Ara Nigogos Kahyaoglu

Permanent Address :

Degree and Date to be conferred : Master of Science in Chemical Engineering, 1989.

Date of Birth :

Place of Birth :

Collegiate Institutions attended :	Dates	Degree	Date
Istanbul University, Turkey	74-78	B.S.	June 78
Istanbul University, Turkey	78-79	M.S.	Oct 79
University de Poitiers, France	79-81	3th Cycle	May 81
New Jersey Institute of Tech.	87-89	M.S. ChE	May 89

Major : Chemical Engineering.

ACKNOWLEDGMENT

The author wishes to acknowledge the helpful discussions and encouragement offered by his adviser, Dr. Edward C. Roche, Jr., also wishes to acknowledge the helpful corrections by Dr. Deran Hanesian and Professor Irwin Hundert.

In addition supports from his wife, Arpi, and his friend Hakan are gratefully acknowledged.

TABLE OF CONTENTS

LISTS OF TABLES AND FIGURES

I. Introduction

PRIDE is an acronym for Plastics Recycling In Defined Environments. PRIDE is a interactive planning tool that models the collection and recycling systems of municipal solid waste, and allows for their comparison. The model has an user-friendly menu driven interface which provides the user with all of the information needed to effectively use it. The underlying purpose of the PRIDE program is to evaluate plastic recycling.

The first version was written in Microsoft Interpreted BASIC (Beginner Assembly Symbolic Instruction Code) which is still a popular programming language and is readily available to the microcomputer purchaser. The second version of PRIDE is coded in the C language. The program consists of eight independent source files (with 61 distinct subprograms) which are combined in a project file for compilation. This program will run on the IBM PC family of computers, and most IBM compatibles.

The BASIC version had many restrictions and was incapable of being modified due to the size restriction. Recoding PRIDE in the C language permitted the inclusion of more efficient and rigorous file handling and checking, a faster execution, better structured variables, the ability to generate printed copy of the data files used during the execution phase, and the ability to modify and expand the program code. A second accomplishment was the achievement of true portability of the source code between different machines and operating systems.

The economic analysis model was originally developed and prepared by the Graduate School of Management, Interfunctional Management Program

Group (August 1987) for CPRR (Center Plastics Recycling Research) at Rutgers the State University of New Jersey.

The large number of calculations makes for logical implementation on a digital computer.

II. BACKGROUND

A. BASIC DATA ON SOLID WASTES

Environmental pollution which nowadays is becoming a major concern sophisticated has been a problem for human beings in developed countries. Public health and ecological impacts, such as air and water pollution, are strongly related to the improper management (storage, collection and disposal) of solid wastes.

Although the data are varied. Recent estimates indicate that an average of 4.4 billion tons of solid wastes (industrial, municipal, agricultural, mineral, animal, ...) are generated each year in the United States. An increase to 12 billion tons will be expected by the year of 2000 [1]. The numbers indicate that this problem will remain one of the most contemporary problems especially in the populated suburban areas [2].

Systems involving recovery of resources from solid wastes depend primarily upon economics and local situations such as characteristics of refuse and site availability for landfilling.

The most important recyclable components are paper, glass and metallic substances. These materials are recoverable and recyclable.

Average Composition of MSW in US (% wt)			
Waste Component	Nemerow, N.L.	CPRR	McMahon,J.
Paper	40.0	42.0	39.0
Glass	10.0	9.0	4.0
Aluminum	0.6-1.0	1.0	1.0
Ferrous	8.0	8.0	2.0
Plastics+Textiles	6.4-10.0	7.0	9.0
Garbage+Yard waste	16.0	16.0	23.0
Others	15.0	17.0	22.0
Total dry Weight	73.0	-	-
Moisture	27.0	-	-

Table 1 : Average Composition of Municipal Solid Waste in US.

CPRR : Center for plastics recycling and research.
Nemerow, N.L. "Industrial Solid Wastes" Bullinger Publishing Co., Cambridge, MA 1984.
McMahon, J. "J. of Waste Age" Dec. 1988 p. 69

In the United States, 40-50 million tons of paper or paper products (highest recovery rates include corrugated boxes, newspapers and office paper) have been used annually. Paper products are among the easiest to recycle and almost 25% of this amount is recycled. Unfortunately recycled paper prices are not currently competitive. Chemicals used for ink removal and mixed paper product collection lower its quality as well as the selling price.

Ferrous metals mainly come from tin cans in the refuse stream. For these, the principal market is the copper-smelting industry. Recent costs associated with separating, recovering and processing this material have not been economical.

Aluminum can utilization as soft drink beverage containers is still increasing. The cost of raw aluminum production is related to the highly expensive use of electrical energy associated with the electrolytic process of aluminum recovery.

Glass has a low salvage value, and industrial people have been resistant to reusing it unless it is of a single color and generic grade. One of the major constraints is metallic impurities. Economic limitations seem to control glass recovery rate and potential for recycling. Silica is not expensive as a raw material, but fuel and energy to fuse it into a product play a vital role on its recovery. One use of residual glass from incinerators is for the manufacture of glass wool. Glass wool is used for home insulation and as an aggregate additive for highway construction and repair.

Most plastics are non-biodegradable, hence deterioration in landfills is practically impossible. Recovery of almost all plastics can be accomplished by electrodynamic separation processes. Another

significant problem is that when some plastics are incinerated they yield hazardous vapor products and thus pollute the atmosphere.

B. TREATMENT METHODS OF SOLID WASTES

There are especially four principal methods of treating Municipal Solid Wastes (MSW).

1. Biodegradation
2. Incineration
3. Landfilling
4. Recycling

Each of them will be discussed briefly in subsequent parts.

Volume and size reduction of the refuse stream is important because any operation such as compaction which reduces the original volume of collected and transported refuse will reduce the total cost as well as the collection time and thus have a significant savings in operating costs. There is no optimal method of municipal solid waste disposal. The choice depends on many circumstances such as environmental pollution, landfill availability, transfer stations, transportation costs, etc... Since the program PRIDE is concerned with recycling with emphasis on plastics, the last alternative method is the focal point of discussion.

1. BIODEGRADATION

Composting is an aerobic process of natural fermentation. The finished dry product of the process (biomass) is organic matter. This process has two primary objectives. The primary one is the inactivation of pathogenic microorganisms . The second one is the production of a final material for further utilization.

Municipal solid wastes which contain a relatively high (C/N) ratio is a source for this disposal means. The carbon is converted to

CO_2 . This biodegradation process is similar to that which occurs naturally in a forest. The aerobic decomposition can be carried out in biological reactors which prevents the formation of odorous products. One of the recent applications of this is the biological treatment of aromatic compounds and their derivatives at low concentrations in water.

2. INCINERATION

Incineration is a high temperature combustion process which converts combustible wastes into inert residue and produces usable energy.

The advantages of incineration are :

a. Less landfill is required
b. Production of a nuisance free ash.
c. Steam production (generation of electric power and low pressure steam for heating of buildings).
d. Climate independency
e. 24-hour rated capacity
f. Flexibility for expansion

The disadvantages of Incineration are :

a. Fly ash production (largely mineral matter and unburned organic matter)
b. Residual ash disposal
c. Air pollution (poisonous gases are given off during combustion of some synthetic polymers)
d. Construction, maintenance and repair costs of an incineration plant are high.

In an incineration system there are three major components :

(1) The furnace is an enclosed, refractory-lined structure, equipped with grates and supplied with excess amounts of air in which the burning process takes place.

(2) The combustion chamber is a secondary unit in which complete combustion of the gaseous products occurs, and where the heavy particles of fly ash settle out.

(3) The subsidence chamber is used to complete the fly ash removal. For a proper level of fly ash removal, wet collectors and scrubbers can be used.

Pyrolysis (heating wastes in airfree chambers at high temperatures as high as 1700°C) is an alterative method which could be cited. It provides some advantages over incineration :

- fewer air pollution problems
- lower capital costs
- self-sufficiency of fuel
- potential for recovering chemicals.

3. SANITARY LANDFILLS

Sanitary Landfills are regions where solid wastes are disposed without creating a nuisance to public health and safety by utilizing the principal's of engineering. In the United States, the largest percentage of disposal of Municipal Solid Wastes is in landfills. However the availability of landfill sites is diminishing and the associated disposal costs are increasing.

At sanitary landfills, the volume of solid waste is minimized by reducing its size and covering it by a thin layer of earth or sand.

Most plastics are non-biodegradable or their biodegradation requires a long period of time. Plastics thus are in general a a landfill problem. Most landfills are far from the solid wastes collection point. The availability of suitable sites at reasonable leasing or purchasing costs, the hauling distance of refuse, the cost of landfill equipment will dramatically affect the usefulness of landfills. For long-distance hauling, baling will be necessary for more economical transportation. Baling also reduces operational problems and disease carrying vectors.

The protection of the health of the public is the main reason for sanitary landfill. The covering of the compacted MSW by a thin layer of soil reduces the presence of carriers and enhances the extermination of flies. One of the main problems associated with landfills is rainwater seeping through the soil layer which may precipitate into the public water streams or aquifer and potentially cause contamination. This seepage can be monitored by parameters such as pH and total bacteria variations.

The soil coverage leads to a biological degradation under anaerobic conditions of the MSW. With oxygen eliminated the process may produce dangerous gas mixtures which can be avoided by proper ventilation. A number of methods are available to prevent these hazards [12]. An alterative is the closure of the landfill by capping it with a PVC (polyvinyl chloride) liner and then covering it with 2 feet of clean sand. This capping scenario has been accomplished at Palm Beach County, Florida [7]. The goal of this operation is the production of methane as a result of the decomposition of the refuse (10 million cu ft of CH_4 per day per 240 acre site). The durability of the methane production process is climate dependent. At the Palm Beach landfill the leachate

is controlled by collecting it via a perimeter collection system and pumped it to a holding tank for subsequent treatment.

Most metallic substances, natural and synthetic rubber, glass, plastics will remain inert in landfill. Even paper has not degraded in some landfills. Hence landfilling is not a viable method for the social problem of MSW. A combination of recycling, resource recovery, incineration and landfilling, plus added environmental protection and mitigation measures will be a solution.

For every ton of recycled wastes that is not landfilled, solid waste tonnage is reduced, landfill costs are lowered, landfill life is extended, and operational savings are achieved. The avoided disposal costs should properly be credited as a benefit of recycling.

4. RECYCLING

Most states are developing Municipal Solid Waste Plans with recovery of recyclable materials [2]. The non-recyclable household refuse would be disposed of in waste-to-energy incinerators, with the residual ash from the incinerators being disposed of in landfills.

Recycling of newspaper, aluminum and glass is well known. Plastics recycling is relatively new and will come under particular scrutiny because of their low weight to volume ratio [3]. Presently, only a small percentage of plastic material has been recycled. The data indicate that plastic material has the potential for being recycled on a nationwide scale [4], [5], [6].

4-1. The recycling systems that have been employed can be classified as :

- Voluntary central buy-back

- Not for reimbursement
- Curbside pick-up.

Any recycling system is successful when the following factors- collection,sorting,reclamation and marketing- are satisfied as summarized in Table 2.

Collection	Sorting	Reclamation	Marketing
The material will be collected.	Sortation into generic type if mixed collection involved.	Quality of recovered material must be enhanced.	Selling of recycled material to end-use markets.

Table 2: Factors for satisfied Recycling System

"The continuing concentration of our population,economic and population growth, improvements in standard of living, continuing technological progress, and marketing have all contributed to a raising tide of scrap, and discarded and waste materials. These have caused communities serious financial, management, intergovernmental, and technical problems in the disposal of these wastes. The problem ..."

Subtitle A

Resource Conservancy and Recovery Act (1976)

4-2. Curbside Collection System

The potentially curbside recyclables are:

- Newspapers
- Aluminum and steel cans
- Glass bottles and jars (clear and colored)
- Plastic beverage containers

The mandatory participation of homeowners in a recycle program optimizes the curbside collection system. The recovery rate is potentially very high when the participants also spend an additional effort by sorting materials into generic types. The size reduction of recyclable materials by the populous by bundling newspapers, crushing cans is desirable for economical and sanitary reasons. The compaction by hand of residential plastics by households examined was by V.R. Sellers[3]. She concluded that the volume of to solid waste stream is not a definitive characteristic such as the weight. Because of spring back plastic material's density can easily change as well as its its original size and shape. The use of a compactor is undesirable when comingled collection is used because glass may be broken during the compaction.

Vehicle type selection and fleet sizing were examined by R.Garrison [8] and found to be two major components in planning and estimating the costs of curbside collection. The most expensive component of the curbside collection system is labor and maintenance.

The waste plastics can follow one of two main paths - disposal or recycling. If disposal is adopted the alternatives are incineration and/or landfill. Since PRIDE is an economic analysis program to

evaluate the collection and sortation of MSW's recyclables including plastics, this will be discussed separately next.

4-3. Reasons For Plastic Recycling

Many of solid-waste disposal sites in use are at or near capacity. Plastic material have little chemical effects in landfills, however they sharply increase the volume of refuse that must be disposed of. The burning of chlorinated plastics generate large amounts of thermal energy. In addition,the combustion process produces vapors of hydrogen chloride, which are toxic. Incineration for energy recovery is a good use for plastics because of their high fuel value. If the costs associated with collection, separation, cleaning and reprocessing are below the cost of virgin resin, plastic recycling is by definition profitable.

4-4. Plastic Recycling

Plastic material are produced from petroleum or natural gas, which are limited resources.

There are two main sources of plastic wastes :

- manufacturers,
- post-consumers.

There are two main types of plastic resins :

- thermoplastics which can be easily reformed by heat,
- thermosets which are difficult to recycle because of their cross-linked bonds that will be broken during melting.

In 1985, the United States produced over 21 billion tons of plastic material. During 1986, 13 billion pounds of plastics were used in packaging in the US [5]. A little over half of the 1986 packaging plastics were consumed in making rigid containers with the remainder utilized in making flexible packaging. PET (polyethyleneterephtalate) is used for soft drink bottles and HDPE (high density polyethylene) is used for milk jugs. These are the primary polymers consumed in beverage containers category. The reclamation effort is concentrated for the recovery of PET and HDPE.

4-4.1. Reclamation Systems

Plastic objects are made up of parts, and these include different structural aspects. In plastic recycling centers, techniques are developed for the isolation of generic polymers and the rejection of aluminum tops, labels, etc.

a. PET dry reclamation system

In this system, paper and polyolefin labels, HDPE, HDPE base cups and aluminum lids are mechanically separated from PET bottle body.

b. Wet sorting system

Shreaded material (gross size reduction) is processed in a grinder to generate small particles. The label is removed by air filtration and the base cup which is made of HDPE is separated from the aluminum lid and PET by water flotation. Further, PET and aluminum are separated by use of electrostatic technology.

The recovered PET/HPDE is refabricated but is not used in food and pharmaceutical products. The recovered plastic is sold to

reclaimers for further processing (carpeting, geotextiles, strapping, fiber, injection and structural foam molding, etc.).

PET Beverage Container Recycling (millions of lbs)			
1979	1982	1984	1986
8	40	110	130
Source : CPRR, Rutgers the State University of NJ			

Table 3 : PET Beverage Container Recycling

PET from recycled bottles is reclaimed reclaimed over 99% and the economics of this process are favorable. For comparison, the prices of virgin PET and HDPE are almost double of reclaimed ones. Hence, the market for collected and reclaimed post-consumer plastics is growing rapidly. With the consumer separating recyclables from their household garbage it is anticipated that this will have a meaningful impact of reducing the MSW that must be disposed of via incineration/landfilling.

5. BRIEF COMPARISON OF TREATMENT METHODS

Solid wastes remain at the point of origin until a decision is made to collect and dispose of them. It was mentioned then some of the means of disposal available are incineration, pyrolysis, landfill, and recycling for resource recovery. Federal and local regulations, geological and seasonal conditions, and overall economics generally determine which method is the most economically acceptable without causing a nuisance to the public health. The most recent indications are that recycling is favored rather over other means of disposal, with landfilling being the dominant form.

C. PROGRAM OVERVIEW

The conversion of PRIDE from the BASIC language to C was deemed necessary because :

1. The size of the original program code prohibits modification, and
2. The BASIC version of the program is not truly portable to other computers.

BASIC is not a portable language; it is machine dependent. Each machine has its own version of BASIC with different extensions and syntaxes. BASIC was created for the first built 8-bit digital computer, and thus is an over grown calculator. The size of the interpreter is 64 Kbytes. Above this limit BASIC creates some complications. BASIC can be expanded by using the changing options, or by using a BASIC compiler.

For the above reasons the best route to follow was to translate the code into a higher level language such as PASCAL, C or FORTRAN. C was chosen because of the ease in handling stream files. C was originally designed for the UNIX operating system and first implemented on the DEC PDP-11 [10]. Its antecedent is the language B which was a " typeless " language and was written by Ken Thompson in 1970 for the first UNIX system on DEC PDP-7.

Most manufacturers and dealers are supporting the standardization of UNIX as an operating system. The C language is a structured, modular, compiled, general purpose intermediate/high level language. It is portable which means that an application program written in C can be transferred to another computer.

Another advantage relates to program development and maintenance. The low price TURBO C compiler of Borland is excellent when compared to its next competitor on single copy basis. TURBO C follows ANSI standardization, and has excellent user interface. Passages from EDIT to COMPILE or from COMPILE to RUN are automatic. TURBO C has a utility library and has a very fast compilation rate (7000 lines per minute) [11].

TURBO C makes extensive syntax checking and thus the user does not have to make any error-listing. Consequently, TURBO C presents a convenient environment, and comes equipped with over 300 library routines-functions and macros that the user can use to perform a wide variety of tasks, such as : low- and high-level input-output (I/O), string and file manipulation, data conversion, etc... [11].

For portability reasons only the inherent functions that UNIX as developed by AT & T were used. The AT & T of C/UNIX conform with the ANSI (American National Standards Institute). Borland's TURBO C has a few embellishments such as the "gotoxy" function. If there is no predefined function such as "gotoxy", then a procedure "gotoxy" which will involve the cursor movement function must be written.

The other problem area is the screen framing procedure "frame_screen". In a slow machine it will slow down the execution. In the "main_menu" this can be totally eliminated by entering the proper value at option (5). The user will thus change the logic of the globally defined variable NOFRAME. The default option is to have the screen will be automatically not framed.

What are the advantages of TURBO C over TURBO PASCAL and FORTRAN ? First of all, both TURBO C and TURBO PASCAL present the same editing

window configuration. The string functions are better defined in C. On the other hand, the first design of PASCAL was as an educational tool. Therefore it has widely extensive structural type checking. This is a good property for structural programming but, at the same time, it is a time consuming item when one wants to develop large programs.

FORTRAN is an old high level language and the compiler is expensive (BORLAND does not have a compiler). The Dynamic Memory Allocation also doesn't exist in FORTRAN.

When designing the program, we incorporated a global variable MAX_INDEX and defined it to be 11 (the 11th being null) to handle the 10 files per data set, so that these would be compatibility with the original BASIC version. After the completion of the data entry for each file in a data set, the user can now generate a hardcopy. The MS-DOS system copy command COPY which is used for the purpose of copying the file to the printer. The syntax is : " COPY filename PRN:" . The COPY command was chosen in favor of the PRINT command in MS-DOS because the PRINT command in MS-DOS has a resident portion that remains in the memory after use. The VMS operating system on a DEC VAX machine could use a COPY command as it is if the printer terminal is made public by the system manager.One can change the COPY command into a PRINT command to the designated printer queue in the VAX so that all users can access a spooled device. The syntax will be : "COPY [uid.dir]filename.ext device:"

The TURBO C compiler forces us to modularized the program because it doesn't handle extremely large source files, due to symbol table limitations.This limitation forces the user to build specific procedures out of the program creating a structured programming.

The STAR file is a "flag file" that indicates which file will in each data set will be used for the next program executions.

The ROCKFORD file keeps track of the currently used data files by the PRIDE. The BASIC version did not provide for any mechanism to check for the consistency of the contents of the ROCKFORD file and the files stored in the directory where the program resides. This capability was added in the C version using the file check menu. This feature is necessary because there is no real file protection mechanisms in MS-DOS.

No backup files are created by PRIDE. This strategy has its good and bad points. Creating a backup in C is easily achieved. After opening and reading the file, close the file, and then use the "COPY filename.ext filename.bak". The problem is if one uses the same route filename but with a different extensions then the backup files will overwrite the previous *.BAK file.

A new feature added was to functional list the specific files being read during the file check. As part of the file check, the procedure checks all of files whether they are the default file or not.

D. PROBLEM AREAS AND DISCUSSION

The prime problem faced converting PRIDE from BASIC to C is that BASIC and C are completely different challengers. BASIC is a line oriented and not easy to convert to a higher level language. The C language version of PRIDE was segmented in specific subprograms and compiled as multiple files to make program 'debugging' easier and quicker. Any TURBO C program developed in an environment as mode provides us with a "make file utility" similar to the "make utility" on the UNIX operating system, which is very useful when working with more than one C file. By changing code in one file, recompilation will be done only on the changed file. Unchanged by referenced source files are not recompiled, that is, the existing object file is used by the linker.

The size of the PRIDE program is approximately 12,000 lines of C code. The executable file is of the range of 160KB.

The fundamental philosophy of the program conversion was to avoid Borland specific functions provided with TURBO C. Sixteen bit arithmetic was arbitrarily used to maintain simplicity, and enhance portability on PC's. Only slight modifications are needed to move the code to Apple McIntosh PC's. All interfaces are done in character mode to avoid the use of graphic libraries which are machine dependent.

We extensively used the Borland "gotoxy" function for cursor movement. For other C compilers this library function can be mimicked.

String operations and character manipulations are more convenient in C than in FORTRAN. In addition, Borland's product line of compilers contain a text editor (similar to Word Star) and are cost effective. Currently Borland does not market a FORTRAN compiler/linker.

The use of structured programming is the preferred form of any simulation program. The C language does not permit multiple entries (also true for PASCAL), as contrasted with FORTRAN.

The interpretative BASIC code was found at the time conversion to have several errors which would not be detected until they were executed. Examples are :

- Line 5060 read two values and wrote five different values on line 5370, and
- Line 22480 did not exist.

In contrast, most errors in a high level language are detected by the compiler.

Extensive checking of file names was added to the revised code so as to be compatible with both MS-DOS and with other operating systems. The use of 8-character file names and 3-character extensions means that compatabilty with most operating systems will be ensured.

BASIC permits variables to be defined and used for multiple purposes. The language C requires that the variable type can not be changed. To be compatible with the BASIC, the zero index of C variables was ignored so as to avoid index referencing. The B[] character array uses the zero entry since the variable use is to manipulate string items. The inherent nature of the C language is "structured programming". This type of programming rarely utilizes statement labels, and thus the familiar "TO GO..." is avoided.

III. RESULTS AND CONCLUSION

The C-language PRIDE code uses the same input/output file set as the BASIC program. Execution of PRIDE yields the same answers as the BASIC program. Using different sets of data the conversion was verified as being complete.

Regional Data from CITY1_1.REG file

The size of the region of interest (square miles)...	2.10
Population density (people/sqr mile).....	5800.00
Route miles (total miles in one collection cycle)...	50.00
Number of collection points.....	4300.00

Policy Data from file CITY1_2.POL

Compliance rate (0-100) for:	
Aluminum	80.00
Newspaper	80.00
Glass	80.00
Plastics	30.00
Other Material	0.00
Pay-out rate for plastics ($/lb)	0.00
Monthly public awareness program cost	42.00
Number of collection cycles per month	4.00
Are recyclables and solid waste collected simultaneously? (1-yes/0-no)	0.00

Solid waste Data from file 30K_A.WAS

Solid Waste Characteristics (screen 1)

(Most values in this module are universal and will not vary across different collection system types.)

Average amount of solid waste created per person per month (lbs/month).....	85.00
Percentage (0-100) of soid waste that is:	
Aluminum.....	0.75
Newspaper.....	12.50
Glass.....	8.70
Plastics.....	1.25
Other material of interest.....	1.50
Percentage (0-100) of plastics that are:	
PE.....	40.00
PET.....	60.00

Solid Waste Characteristics (screen 2)

Percentage (0-100) of PET that is:	
Clear.....	70.00
Colored.....	30.00
Percentage (0-100) of glass that is:	
Clear.....	50.00
Green.....	25.00
Brown.....	25.00
Average density (lb/ft^3) of solid waste....	24.40
Average density (lb/ft^3) of recyclable:	
Aluminum.....	2.09
Newspaper.....	21.28
Glass.....	29.79
Plastic.....	1.27
Other Material.....	6.10

Solid waste Data from file CITY1_2.WAS

Solid Waste Characteristics (screen 1)

(Most values in this module are universal and will not vary across different collection system types.)

Average amount of solid waste created per person per month (lbs/month).....	100.00
Percentage (0-100) of soid waste that is:	
Aluminum.....	1.00
Newspaper.....	23.00
Glass.....	7.00
Plastics.....	9.50
Other material of interest.....	0.00
Percentage (0-100) of plastics that are:	
PE.....	50.00
PET.....	50.00

Solid Waste Characteristics (screen 2)

Percentage (0-100) of PET that is:	
Clear.....	50.00
Colored.....	50.00
Percentage (0-100) of glass that is:	
Clear.....	34.00
Green.....	33.00
Brown.....	33.00
Average density (lb/ft^3) of solid waste....	22.00
Average density (lb/ft^3) of recyclable:	
Aluminum.....	2.80
Newspaper.....	25.00
Glass.....	22.00
Plastic.....	3.00
Other Material.....	0.00

Collection Equipment Data from CITY1_1.EQP file

Collection Equipment Vehicle Type	1	2	3	4
# of Units	2.00	0.00	0.00	0.00
Weight Capacity (lbs)	6580.00	0.00	0.00	0.00
Volume Capacity (ft^3)	300.00	0.00	0.00	0.00
Equipment Cost /month/unit	367.00	0.00	0.00	0.00
Overhead /month/unit	75.00	0.00	0.00	0.00
Cost/mile/unit	0.81	0.00	0.00	0.00
Staff/unit (frac ok)	1.00	0.00	0.00	0.00
Avg # Collect points/month/unit	8600.00	0.00	0.00	0.00

Labor Costs Data from CITY1_1.LAB file

Labor Costs
The average monthly cost per laborer for:

Collectors.....	2500.00
Material Handlers.....	2500.00

Material Handling Costs Data from CITY1_1.MAT file

Material Handling

	# operators	# machines	equip cost ($/month/unit)
Sorting-manual	0.00	n/a	n/a
Granulation/Shredding	0.00	0.00	0.00
Baling	0.00	0.00	0.00
Crushing (glass/cans only)	0.00	0.00	0.00

Monthly overhead for the material handling facility.... 0.00

Shipping Costs (screen 1) Data from 30K_A.SHP file

Distance (miles) from the materials handling facility to:	
Glass market	0.00
Newspaper market	0.00
Aluminum market	0.00
Plastic processing plant	0.00
Landfill	0.00
Market for other materials	0.00
Cost ($) per mile per shipment for:	
Glass	0.00
Newspaper	0.00
Aluminum	0.00
Plastic	0.00
Solid waste	0.00
Other material	0.00

Shipping Costs (screen 2)

Weight capacity (lbs) per shipment of:	
Glass	40000.00
Newspaper	0.00
Aluminum	0.00
Plastic	0.00
Solid waste	0.00
Other material	40000.00

Shipping Costs (screen 1) Data from CITY1_1.SHP file

Distance (miles) from the materials handling facility to:	
Glass market	9.00
Newspaper market	9.00
Aluminum market	9.00
Plastic processing plant	0.00
Landfill	10.00
Market for other materials	0.00
Cost ($) per mile per shipment for:	
Glass	0.81
Newspaper	0.81
Aluminum	0.81
Plastic	0.00
Solid waste	0.00
Other material	0.00

Shipping Costs (screen 2)

Weight capacity (lbs) per shipment of:	
Glass	3000.00
Newspaper	3000.00
Aluminum	500.00
Plastic	500.00
Solid waste	0.00
Other material	0.00

Plastics Processing Costs Data from CITY1_1.PLA file

Equipment cost/month.....	0.00
Direct process costs ($/ton).....	0.00
Processing facility monthly overhead..	0.00
# of operators at processing plant....	0.00
Process efficiency ratio (0-1).....	0.00

Disposal Costs Data from CITY1_2.DIS file

Disposal Costs	
Tipping fee ($/ton).....	75.00
Government subsidy or abatement ($/ton)..	0.00

Selling Prices Data from CITY1_2.SEL file

Selling Prices of Recycled Waste Material

Price per lb of:	
PE	0.00
Clear PET	0.00
Colored PET	0.00
Clear glass	0.00
Green glass	0.00
Brown glass	0.00
Aluminum	0.10
Newspaper	0.00
Mixed PE/PET	0.04
Mixed glass	0.10
Other material of interest	0.00

Are you selling (0)-mixed or (1)-sorted PE/PET...	0
Are you selling (0)-mixed or (1)-sorted glass....	0

Collection Point Data from CITY1_1.COL file

Collection Point Data	
Equipment cost per collection point..	0.00
Overhead per collection point.....	0.08

Report from output file OUT1.RPT

Revenue, Subsidies, and Credits ..... $/month

Plastics	1389	
Glass	6821	
Aluminum	974	
Newspaper	1121	
Other Material	0	
Tipping Fee Credit (wt based)	12629	
Government Subsidies	0	
Total Revenue		22933
Tipping Fee Credit (volume based)	22371	
Adjusted Total (not used in subsequent calculations)		32675

Recycling Costs ... $/month

Collection Point	400	
Collection	6046	
Material Handling	0	
Shipping	852	
Plastics Processing	0	
Total Recycling Costs		7299

Nonrecyclable Costs ... $/month

Collection	0	
Material Handling	0	
Shipping	0	
Tipping Fees	0	
Total Nonrecyclable Cost		0

Net Revenue ... $/month 15635

Break Even Tipping Fee...$/ton (Net Revenue = $0.00) -18

Costs Breakdown ... $/month

	Plastic	Glass	Aluminum	Newspaper	Other
Collection Point	246	66	74	14	0
Collection System	2580	691	776	1999	0
Material Handling	0	0	0	0	0
Shipping	0	166	142	545	0
Processing	0	0	0	0	0
Total	2826	923	993	2557	0

Plastic Collection Cost/lb 0.08

	Wt.(lbs)	Vol.(yd^3)
Total Waste Stream per month	1218000	2051
Recycled Plastic	34713	429
Recycled Glass	68208	115
Recycled Newspaper	224112	332
Recycled Aluminum	9744	129
Other Recycled Material	0	0
Total recyclables removed from waste stream	336777	1004
% of total waste stream	28%	49%

IV. REFERENCES

1. Nemerow, N. L., Industrial Solid Wastes, Bullinger Publishing Co. , Cambridge, MA 1984.
2. The New York Times, Tues.March 14, Thur.March 16 1989.
3. Sellers, V.R. , Franklin Associated Ltd. NJ.
4. Marrow, D.R., Amini,M.A., Adams,J.C., and Merriam, C.N., Converting and Packaging, Dec. 1987.
5. Marrow, D.R. , Pearson, W.E. , and Weis, R.S. Paper submitted to New Jersey Commission on Science and Technology Center, Mar 1988.
6. Curlee, R.T. , Conservation and Recycling, Vol 9, No 4, 1986.
7. Snow, D. , The Magazine of Disposal Options December 1988, p. 6.
8. Garrison, R. , Resource Recycling Aug.(1988). Press, Baltimore and London.
9. McMahon, J.Waste Age Dec.(1988) p.69.
10. Kernigan, B.W.and Ritchie, M.D. , The C programming Language, Prentice-Hall, Engelwood Cliffs, NJ.
11. BORLAND's TURBO C, Scotts Valley, CA.
12. Mead, B.E.and Wilkie, W.G.Leachate Prevention and Control from Sanitary Landfills, Paper 35b, 68th National AIChE Meeting, Houston, Texas, Mar. 1971.

Appendix A

PROGRAM LIST

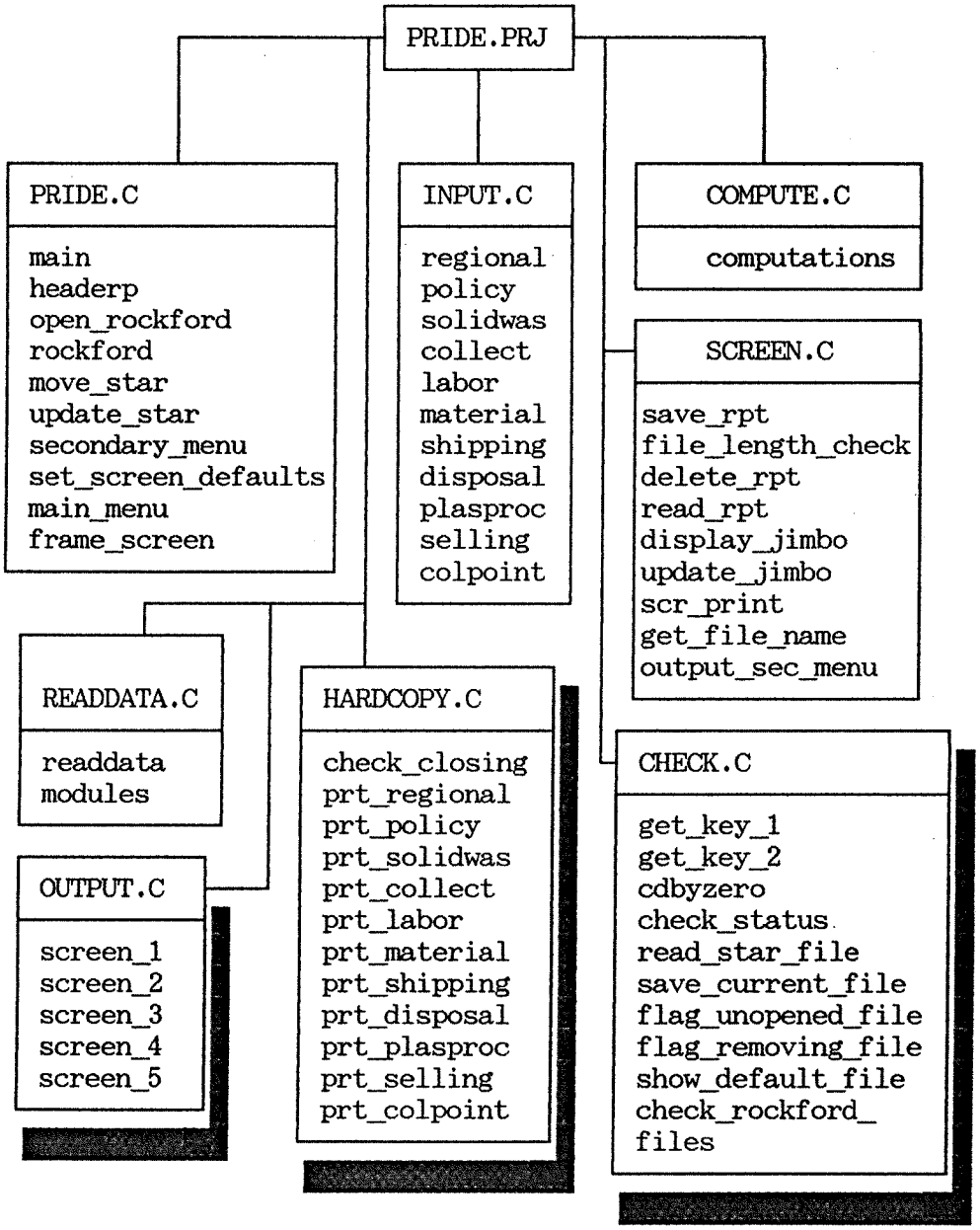

Figure 2: Contents of the eight files containing the C source code for the PRIDE.
Note that the main is contained in PRIDE.C

```
/* FILE : VARIABLE.h */
#include <stdio.h>
#include <conio.h>
#include <dos.h>
#include <io.h>
#include <string.h>
#include <fcntl.h>
#include <sys\stat.h>
#include <ctype.h>
#include <stdlib.h>

/* MS/DOS specific */
/* Constant declarations */
#define VDL  186    /* vertical double line     */
#define URDL 187    /* upper right double line  */
#define BRDL 188    /* bottom right double line */
#define BLDL 200    /* bottom left double line  */
#define ULDL 201    /* upper left double line   */
#define HDL  205    /* horizontal double line   */
#define DEF_NOFRAME 1
#define DEF_DELAY 1
#define MAX_INDEX 11
#define MFL 13
#define ROCKFORD "ROCKFORD"

#if !defined( MAIN )
extern int DELAY;
extern int NOFRAME;
extern int MODULE;
extern int NEW_INPUT_FILE;
extern FILE *star,*fil,*jim;
extern float W[20];
extern float C[33];
extern float DIS[3];
extern float H[12];
extern float S[19];
extern float PR[14];
extern float RE[16];
extern float PT[3];
extern char  *D[12][MAX_INDEX];
extern char  *F[31];
extern int   STAR[12][MAX_INDEX];
extern int   INDEX;
extern int   JIMBO_INDEX;
extern int XX[16];
extern int YY[16];
extern float R[5];
extern float P[10];
extern char B[MFL];
extern float NC[29];
extern float L[3];
extern float PP[6];
extern float CP[5];
extern float CS[7];
```

```
extern float MH[5];
extern float SH[18];
extern float RM[17];
extern float DI[5];
extern float PL[3];
extern float CR[10];
extern float VOLTOT,VOLA,VOLG,VOLO,VOLN,VOLP,VOLW;
extern float COTIP,COPO,BETF;
extern float v;
#else
int DELAY;
int NOFRAME;
int MODULE;
int NEW_INPUT_FILE;
FILE *star,*fil,*jim;
float W[20];
float C[33];
float DIS[3];
float H[12];
float S[19];
float PR[14];
float RM[17];
float RE[16];
float PT[3];
char  *D[12][MAX_INDEX];
char  *F[31];
int   STAR[12][MAX_INDEX];
int   INDEX;
int   JIMBO_INDEX;
int XX[16];
int YY[16];
float R[5];
float P[10];
char B[MFL];
float NC[29];
float L[3];
float PP[6];
float CP[5];
float CS[7];
float MH[5];
float SH[18];
float RM[17];
float DI[5];
float PL[3];
float CR[10];
float VOLTOT,VOLA,VOLG,VOLO,VOLN,VOLP,VOLW;
float COTIP,COPO,BETF;
float v;
#endif

/* prototypes for functions as per ANSII */

void main_menu(void);
void headerp(void);
```

```
void regional(void);
void modules(void);
void policy(void);
void rockford(void);
void solidwas(void);
void collect(void);
void labor(void);
void shipping(void);
void material(void);
void disposal(void);
void plasproc(void);
void selling(void);
void colpoint(void);
int get_key_1(void);
int get_key_2(void);
void readdata(void);
float computations(void);
void screen_1(void);
void screen_2(void);
void screen_3(void);
void screen_4(void);
void screen_5(void);
void cdbyzero(void);
void output_sec_menu(void);
void delete_rpt(void);
int file_length_check(void);
void frame_screen(void);
void write_output(void);
void wait_and_go(void);
void save_current_file(void);
void open_rockford(void);
void secondary_menu(void);
void scr_print(void);
void move_star(void);
void read_star_file(void);
void update_star(void);
void save_rpt(void);
void display_jimbo(void);
void flag_unopened_file(char *pointer);
void flag_removing_file(char *pointer);
void update_jimbo(void);
void read_rpt(void);
void check_rockford_files(void);
void prt_regional(void);
void prt_policy(void);
void prt_solidwas(void);
void prt_collect(void);
void prt_labor(void);
void prt_material(void);
void prt_shipping(void);
void prt_disposal(void);
void prt_plasproc(void);
void prt_colpoint(void);
void prt_selling(void);
```

```
void set_screen_defaults(void);
void show_default_file(char *pointer);
void check_status( int sts );
void get_file_name(void);
void check_closing(int t);
```

```
/* FILE:PRIDE.C */

#define MAIN
#include "variable.h"

void main()
{
  set_screen_defaults();
  window(1,1,80,25);  /* Initialize the screen addressing functions */
  headerp();          /* Displays title pages */
  read_star_file();
  main_menu();
}

void headerp()
{ /*prints header */
  int i;
  clrscr();
  gotoxy(18,5);
  printf("Plastics Recyling In Defined Environments");
  gotoxy(25,8);
  printf("%c",ULDL);
  for(i=0;i<25;i++)printf("%c",HDL);
  printf("%c",URDL);
  for(i=0;i<5;i++)
  {
     gotoxy(25,9+i);printf("%c",VDL);
     gotoxy(51,9+i);printf("%c",VDL);
  }
  gotoxy(25,14);
  printf("%c",BLDL);
  for(i=0;i<25;i++)printf("%c",HDL);
  printf("%c",BRDL);
  gotoxy(30,10);
  printf("P   R   I   D   E");
  gotoxy(34,12);
  printf("Ver.  2.0");
  gotoxy(15,16);
  printf("Center for Plastics Recycling Research (CPRR)");
  gotoxy(16,17);
  printf("Rutgers,The State University of New Jersey");
  gotoxy(27,18);
  printf("Piscataway,NJ 08855");
  gotoxy(56,19);
  printf("By Ara Kahyaoglu");
  gotoxy(58,20);
  printf("March 1989");
  frame_screen();
  sleep(DELAY);
  return;
}

void open_rockford ()
{
```

```
int m,k,j;
FILE *rock;
clrscr();
rock = fopen(ROCKFORD,"r");
if ( rock == 0 )
/* if rock equal to zero */
{
  printf("Can't find ROCKFORD file");
  printf("\n Open a new one ?    ");
  m = get_key_1();
  if( m == -89 )
  {
    rock = fopen(ROCKFORD,"w");
    for(m=1;m<12;m++)
      for (k=1;k<MAX_INDEX;k++)
    {
      fprintf(rock,"\n");
    };
    fclose(rock);
    read_star_file();
    for(j=1;j<MAX_INDEX;j++)
    {
      if(D[MODULE][j]!=0)
      {
        if (STAR[MODULE][j]==1) INDEX=j;
      }
    }
    return;
  }
  else
  {
    sleep(DELAY);
    main_menu();
  }
};
/* if rock is not equal to zero */
for(m=1;m<12;++m)
  for (k=1;k<MAX_INDEX;++k)
{
  for(j=0;j<MFL;j++)B[j]=0;
  fgets(B,MFL,rock);
  file_length_check();
  D[m][k]=malloc(strlen(B));
  strcpy(D[m][k],B);
};
fclose(rock);
read_star_file();
for(j=1;j<MAX_INDEX;j++)
{
  if(D[MODULE][j]!=0)
  {
    if (STAR[MODULE][j]==1) INDEX=j;
  }
  else break;
```

```
  }
}

void rockford()
{ /* File retrieval uses and find the value of MODULE   */
  int j;
  open_rockford () ;
  frame_screen();
  /*switch is a branch statement with valid*/
  /*value of 1,2,...11 */
  switch(MODULE)
  {
    case 1 :
             gotoxy(30,1);
             printf(" Regional Data Files ");
             break;
    case 2 :
             gotoxy(30,1);
             printf(" Policy Data Files ");
             break;
    case 3 :
             gotoxy(27,1);
             printf(" Solid Waste Characteristics Files ");
             break;
    case 4 :
             gotoxy(18,1);
             printf(" Collection Equipment Characteristics Files ");
             break;
    case 5 :
             gotoxy(30,1);
             printf(" Labor Costs Files ");
             break;
    case 6 :
             gotoxy(27,1);
             printf(" Material Handling Files ");
             break;
    case 7 :
             gotoxy(29,1);
             printf(" Shipping Costs Files ");
             break;
    case 8 :
             gotoxy(29,1);
             printf(" Disposal Costs Files ");
             break;
    case 9 :
             gotoxy(23,1);
             printf(" Plastics Processing Costs Files ");
             break;
    case 10:
             gotoxy(15,1);
             printf(" Selling Prices of Recycled Waste Material Files ");
             break;
    case 11:
             gotoxy(25,1);
```

```
                printf(" Collection Point Data Files ");
                break;
        default:
                break;
    }
    for(j=1;j<MAX_INDEX;j++)
    {
      if(D[MODULE][j]!=0)
      {
        gotoxy(35,j+2);
        printf("%s",D[MODULE][j]);
        if (STAR[MODULE][j]==1)
        {
          INDEX=j;
          gotoxy(50,j+2);
          printf("*");
        }
      }
      else break;
    }
    /*print secondary menu */
    gotoxy(18,3+j);
    printf("Available options:");
    gotoxy(20,4+j);
    printf("(1)  Edit an old file");
    gotoxy(20,5+j);
    printf("(2)  Create a new file");
    gotoxy(45,4+j);
    printf("(3)  Destroy a file");
    gotoxy(45,5+j);
    printf("(4)  Return to Secondary menu");
    gotoxy(20,6+j);
    printf("Esc  Return to Main Menu");
    gotoxy(20,7+j);
    printf("(Hit space bar to move file flag ... '*')");
    gotoxy(20,8+j);
    printf("NOTE: Enter from the keyboard: 1, 2, 3, 4, Esc or SpaceBar");
    gotoxy(20,9+j);
    printf("NOTE: If a file is to be printed use option (1) or (2)");
    gotoxy(20,10+j);
    printf("       to print after editing or creation");
    gotoxy(20,11+j);
    secondary_menu();
    return;
}

void move_star()
{
    int i,j,k;
    /* find how many entries in rockford */
    for(j=1 ; (D[MODULE][j]!=0) ; j++)k=j;
    for(i=1 ; i<MAX_INDEX ; i++)
    {
      if (STAR[MODULE][i]==1)
```

```
    {
      gotoxy(50,i+2);
      printf(" ");
      STAR[MODULE][i]=0;
      if ( i >= k ) k = 1;
      else k = i+1;
      gotoxy(50,k+2);
      INDEX=k;
      printf("*");
      STAR[MODULE][k]=1;
      i=MAX_INDEX;
      update_star();
      return;
    }
  }
}

void update_star()
{ /* update the star file */
  int m,k;
  star=fopen("STAR","w");
  if ( star == 0 )
  {
    printf("Can't find STAR file");
    sleep(DELAY);
    return;
  };
  for (m=1;m<12;m++)
    for (k=1;k<MAX_INDEX;k++)
      fprintf(star,"%d\n",STAR[m][k]);
  fclose(star);
  return;
}

void secondary_menu()
{ /*Secondary menu associated with data changes,etc...*/
  int c,h,j,m,k;
  char A[20]; /* buffer for the command to be passed to OS */
  FILE *rock;
  NEW_INPUT_FILE=0;
  c = get_key_1();
  if ( c == -27 )
  {/* User pressed Esc Key
      save star values and return to main menu */
      clrscr();
      main_menu();
      return;
  }
  if ( c == -32 )
  {/* User has pressed the space bar to the star flag
      save star values and return to secondary_menu */
      move_star();
      secondary_menu();
      return;
```

```
    }
    if ( c == -13 )
    {/* User has pressed the carriage return use the default file
        name indicated by the star flag */
        return;
    }
    gotoxy(5,20);
    switch(c)
    {
      case 1 : /* Edit old files */
               printf("What file would you like to edit? ");
               h = get_key_2();
               if ( h == -13 ) /* <CR>, Use Default flagged file */
               {
                 printf("Using %s\n",D[MODULE][INDEX]);
                 sleep(DELAY);
               }
               else /* See if the file exist in our recollection */
               { /* || OR */
                 if (( h == -1 )||( h == 0 )) file_length_check();
                 for(j=1;(D[MODULE][j]!=0);j++)
                   if (strcmp(D[MODULE][j],B)==0) h = j;
                 if(j>(MAX_INDEX-1))
                 {
                   printf("You have reached the maximum number of files for
this data set\n");
                   sleep(DELAY);
                 }
                 else
                 { /* move the star flag to found file */
                   STAR[MODULE][INDEX] = 0 ;
                   INDEX = h ;
                   STAR[MODULE][INDEX] = 1 ;
                   update_star();
                 }
                 sleep(DELAY);
               } ;
               for(j=0;j<MFL;j++)B[j]=0;
               break;
      case 2 : /* Edit new file  */
               printf("Enter new file name : ");
               h = get_key_2();
               if (( h == -1 )||( h == 0 ))
               { /* h has the length of string B if h > 0 */
                 if (( h = file_length_check() ) > 0 )
                 {
                   for(j=1;(D[MODULE][j]!=0);j++);
                   if(j>(MAX_INDEX-1))
                   {
                     printf("You have reached the maximum number of files
for this data set\n");
                     sleep(DELAY);
                   }
                   D[MODULE][j] = malloc(h) ;
```

```
                    if ( D[MODULE][j] == 0 )
                    { /* check for success */
                      printf("Cannot allocate memory for the file name
%s\n",B);
                      sleep(DELAY);
                    };
                    strcpy ( D[MODULE][j],B );
                    /* update the names file */
                    rock=fopen(ROCKFORD,"w");
                    if ( rock == 0 )
                    {
                      printf("Can't find ROCKFORD file");
                      sleep(DELAY);
                      return;
                    };
                    for(m=1;m<12;m++)
                      for (k=1;k<MAX_INDEX;k++)
                        fprintf(rock,"%s\n",D[m][k]);
                    fclose(rock);
                    /* create the file B */
                    rock=fopen(B,"w");
                    if ( rock == 0 )
                    {
                      printf("Can't find %s file",B);
                      sleep(DELAY);
                      return;
                    };
                    fclose(rock);
                    for(j=1;(D[MODULE][j]!=0);j++)
                      if (strcmp(D[MODULE][j],B)==0) h = j;
                    if(j>(MAX_INDEX-1))
                    {
                      printf("You have reached the maximum number of files
for this data set\n");
                      sleep(DELAY);
                    }
                    else
                    { /* move the star flag to found file */
                      STAR[MODULE][INDEX] = 0 ;
                      INDEX = h ;
                      STAR[MODULE][INDEX] = 1 ;
                      update_star();
                    }
                  }
                }
                for(j=0;j<MFL;j++)B[j]=0;
                sleep(DELAY);
                NEW_INPUT_FILE=1;
                break;
    case 3 : /* Delete a file  */
                printf("What file would you like to delete? ");
                h = get_key_2();
                if (( h == -1 )||( h == 0 ))
                { /* h has the length of string B if h > 0 */
```

```
                if (( h = file_length_check() ) > 0 )
                {
                  h = 0 ;
                  for(j=1;(D[MODULE][j]!=0);j++)
                    if (strcmp(D[MODULE][j],B)==0) h = j;
                  if(j>(MAX_INDEX-1))
                  {
                    printf("You have reached the maximum number of files
for this data set\n");
                    sleep(DELAY);
                  }
                  else
                  { /* if star flag was set move it to the previous file
name*/
                    if(( STAR[MODULE][h] == 1 )&&( h != 1 ))
                    {
                      STAR[MODULE][h]=0;
                      STAR[MODULE][h-1]=1;
                    }
                  }
                  /* Get rid of the file name */
                  gotoxy(5,21);
                  printf("File to be deleted : %s\n",D[MODULE][h]);
                  sleep(DELAY);
                  D[MODULE][h] = 0;
                  if ( j > h ) /* now there is hole in the names array */
                    for ( k = h ; k < j ; k++ )
D[MODULE][k]=D[MODULE][k+1];
                  /* update the names file */
                  rock=fopen(ROCKFORD,"w");
                  if ( rock == 0 )
                  {
                    printf("Can't find ROCKFORD file");
                    sleep(DELAY);
                    return;
                  };
                  for(m=1;m<12;m++)
                    for (k=1;k<MAX_INDEX;k++)
                      fprintf(rock,"%s\n",D[m][k]);
                  fclose(rock);
                  /* delete the file B */
                  sprintf(A,"DEL %s",B);
                  system(A);
                  update_star();
                  rockford();
                }
              }
              for(j=0;j<MFL;j++)B[j]=0;
              sleep(DELAY);
              break;
    case 4 : /* Save star values and return to secondary menu */
              update_star();
              sleep(DELAY);
              modules();
```

```
                break;
        default:
                printf("* * * Un-expected input * * *");
                sleep(DELAY);
                gotoxy(5,20);
                printf("* * * Re-enter selection * * *");
                sleep(DELAY);
                secondary_menu();
                break;
    }
    return;
}

void set_screen_defaults()
{
    int c;
    FILE *def;
    NOFRAME = DEF_NOFRAME;
    DELAY   = DEF_DELAY;
    def = fopen("SCREEN.DEF","r");
    if ( def == NULL )
    {
      clrscr();
      frame_screen();
      def = fopen("SCREEN.DEF","w");
      if ( def == NULL )
      {
        printf("Cannot open SCREEN.DEF file\n");
        sleep(DELAY);
      };
      gotoxy(10,10);
      printf("Enter Screen Display Delay in seconds [1 second] ");
      /*get_key_2 stores # in v */
      c=get_key_2();
      if( c ==   0 ) DELAY = v;
      if( c == -13 ) DELAY = DEF_DELAY;
      if((DELAY < 0) || (DELAY > 10 ))
      {
        printf("\nYou might fall asleep if delay is %d\n",DELAY);
        DELAY = 2;
        printf("\nSo I will use the default of %d seconds\n",DELAY);
      };
      gotoxy(10,15);
      printf("Do you want to frame the screen ( Y/N [N]) ");
      c=get_key_1();
      if( c == -89 ) NOFRAME = 0;
      fprintf(def,"%2 \n%2d \n", NOFRAME, DELAY );
    }
    else
    {
      fscanf(def,"%d %d", &NOFRAME, &DELAY );
    };
    fclose(def);
    return;
```

```
}

void main_menu()
{
  int c;
  clrscr();
  frame_screen();
  gotoxy(35,3); printf("MENU");
  gotoxy(20,7); printf(" (1)  Run Program");
  gotoxy(20,8); printf(" (2)  Enter or Change Parameters");
  gotoxy(20,9); printf(" (3)  Access Output Files");
  gotoxy(20,10);printf(" (4)  File Check");
  /* This is used if files listed in the program have been deleted */
  /* outside the program */
  gotoxy(20,11);printf(" (5)  Install Screen Configuration.");
  gotoxy(20,12);printf(" (6)  Quit ... return to operating system");
  gotoxy(20,16);printf(" NOTE: Options (2) and (3) have secondary
menus");
  gotoxy(20,17);printf(" Enter from the keyboard 1, 2, ... ... , or 6");
  gotoxy(22,14);
  /* ask what do we want to do */
  c = get_key_1();
  switch(c)
  {
    case 1 : /* Read rockford                */
             MODULE=1;
             open_rockford();
             readdata();
             computations();
             main_menu();
             break;
    case 2 : /* Print modules screen to make selection */
             modules();
             break;
    case 3 : /* Access output files on jimbo */
             output_sec_menu();
             main_menu();
             break;
    case 4 : /* File check                   */
             open_rockford();
             check_rockford_files();
             readdata();
             main_menu();
             break;
    case 5 : /* Change the screen configuration file SCREEN.DEF
                start by deleting the file so that pride
                is forced to create a new one
             */
             if(unlink("SCREEN.DEF") == -1 )
             {
               printf("Problem deleting the SCREEN.DEF file via
unlink\n");
               sleep(DELAY);
             }
```

```
                else
                {
                  set_screen_defaults();
                  main_menu();
                };
                break;
    case 6 : /* system */
                clrscr();
                exit(0);
                break;
    default:
                gotoxy(1,1);
                printf("Input Error %d",c);
                sleep(DELAY);
                main_menu();
                break;
  }
  return;
}

void frame_screen()
{ /* Make the screen look pretty and abuse the fact this runs on a
     25 MHZ PC-386
  */
  int i;
  if( NOFRAME==1 ) return ;
  else
  {
    gotoxy(1,1);
    printf("%c",ULDL);
    for(i=0;i<78;i++)printf("%c",HDL);
    printf("%c",URDL);
    for(i=0;i<23;i++)
    {
      gotoxy( 1,2+i);printf("%c",VDL);
      gotoxy(80,2+i);printf("%c",VDL);
    };
    gotoxy(1,24);
    printf("%c",BLDL);
    for(i=0;i<78;i++)printf("%c",HDL);
    printf("%c",BRDL);
  };
  return;
}
```

```
/* FILE : INPUT.C */

#include "variable.h"

void regional()
{ /* REGIONAL DATA */
  int c,i,n,h;
  rockford();
  fil = fopen(D[MODULE][INDEX],"r");
  if ( fil == 0 ) flag_unopened_file(D[MODULE][INDEX]);
  for(i=1;i<5;i++)
  {
    R[i]=0.0;
    fgets(B,MFL,fil);
    sscanf(B,"%f",&R[i]);
  };
  fclose(fil);
  clrscr();
  frame_screen();
  show_default_file(D[MODULE][INDEX]);
  gotoxy(33,1);printf(" Regional Data ");
  gotoxy(5,4);
  printf("The size of the region of interest (square
miles)...%12.2f",R[1]);
  gotoxy(5,7);
  printf("Population density (people/sqr
mile).............%12.2f",R[2]);
  gotoxy(5,10);
  printf("Route miles (total miles in one collection
cycle)...%12.2f",R[3]);
  gotoxy(5,13);
  printf("Number of collection
points.....................%12.2f",R[4]);
  XX[1]=XX[2]=XX[3]=XX[4]=57;          /* This routine for inputting data
*/
  YY[1]=4;YY[2]=7;YY[3]=10;YY[4]=13;   /* is repeated for each module.
Only */
  n=0;                                 /* the cursor locations and amt of
*/
  for(;;) /* infinite loop */                    /* data differ.
*/
  { n++;
    if (n == 5)  break;
    gotoxy( XX[n],YY[n]);
    h=get_key_2();
    if (h == -27) break;
    else if (h==0) R[n]=v;
    gotoxy( XX[n],YY[n]);
    printf("          ");
    gotoxy( XX[n],YY[n]);
    printf("%12.2f",R[n]);
    if (h == -85)
      if (n ==  1 ) n=0;
      else n -=2;
```

```
  };
  gotoxy(5,15); printf("                          ");
  gotoxy(5,15); printf("Update changes (Y/N) ? ");
  c = get_key_1();
  if (c==-89)
  { /* Save the record */
    fil = fopen(D[MODULE][INDEX],"w");
    if ( fil == 0 ) flag_unopened_file(D[MODULE][INDEX]);
    for(i=1;i<5;i++) fprintf(fil,"%f\n",R[i]);
    save_current_file();
  }
  else
  {
    if ( NEW_INPUT_FILE == 1) unlink( D[MODULE][INDEX] ) ;
    main_menu();
  };
  gotoxy(5,15); printf("                              ");
  gotoxy(5,15); printf("Print Hardcopy (Y/N) ? ");
  gets(B);
  B[0]=toupper(B[0]);
  if(B[0]=='Y')   /*Do not save but print the data */
  {
    gotoxy(5,15);printf("                            ");
    gotoxy(5,15);printf("Is printer ready (Y/N) ? ");
    gets(B); B[0]=toupper(B[0]);
    if(B[0]=='Y') prt_regional();
  };
  modules();
  return;
}

void policy()
{ /* POLICY DATA */
  int c,i,n,h;
  rockford();
  fil = fopen(D[MODULE][INDEX],"r");
  if ( fil == 0 ) flag_unopened_file(D[MODULE][INDEX]);
  for(i=1;i<10;i++)
  {
    P[i]=0.0;
    fgets(B,MFL,fil);
    sscanf(B,"%f",&P[i]);
  };
  clrscr();
  frame_screen();
  show_default_file(D[MODULE][INDEX]);
  gotoxy(35,1);
  printf("Policy Data");
  gotoxy(5,3);
  printf("Compliance rate (0-100) for:");
  gotoxy(30,5);
  printf("Aluminum.....%12.2f",P[1]);
  gotoxy(30,7);
  printf("Newspaper.....%12.2f",P[2]);
```

```
 gotoxy(30,9);
 printf("Glass.....%12.2f",P[3]);
 gotoxy(30,11);
 printf("Plastics.....%12.2f",P[4]);
 gotoxy(30,13);
 printf("Other Material...%12.2f",P[5]);
 gotoxy(5,16);
 printf("Pay-out rate for plastics ($/lb).....%12.2f",P[6]);
 gotoxy(5,18);
 printf("Monthly public awareness program cost.....%12.2f",P[7]);
 gotoxy(5,20);
 printf("Number of collection cycles per month.....%12.2f",P[8]);
 gotoxy(5,22);
 printf("Are recyclables and solid waste ");
 gotoxy(5,23);
 printf("collected simultaneously? (1-yes/0-no)....%12.0f",P[9]);
 XX[1]=XX[2]=XX[3]=XX[4]=XX[5]=XX[6]=XX[7]=XX[8]=XX[9]=47;
 YY[1]=5;YY[2]=7;YY[3]=9;YY[4]=11;YY[5]=13;
 YY[6]=16;YY[7]=18;YY[8]=20;YY[9]=23;
 n=0;
 for(;;)
 {
   n++;
   if (n == 10)  break;
   gotoxy( XX[n],YY[n]);
   h=get_key_2();
   if (h == -27) break;
   else if (h==0) P[n]=v;
   gotoxy( XX[n],YY[n]);
   printf("           ");
   gotoxy( XX[n],YY[n]);
   if(n==9)printf("%12.0f",P[n]);
   else printf("%12.2f",P[n]);
   if (h == -85)
      if (n == 1) n=0;
      else n-=2;
 };
 if ( P[9] != 1.0 ) P[9] = 0.0;
 gotoxy(5,24); printf("                          ");
 gotoxy(5,24); printf("Update changes (Y/N) ? ");
 c = get_key_1();
 if ( c == -89 )
 { /* Save the record */
   fil = fopen(D[MODULE][INDEX],"w");
   if ( fil == 0 ) flag_unopened_file(D[MODULE][INDEX]);
   for(i=1;i<10;i++) fprintf(fil,"%f\n",P[i]);
   save_current_file();
 }
 else
 {
   if ( NEW_INPUT_FILE == 1) unlink( D[MODULE][INDEX] ) ;
   main_menu();
 };
 gotoxy(5,24); printf("                              ");
```

```
   gotoxy(5,24); printf("Print Hardcopy (Y/N) ? ");
   gets(B);
   B[0]=toupper(B[0]);
   if(B[0]=='Y')   /*Do not save but print the data */
   {
     gotoxy(5,24); printf("                          ");
     gotoxy(5,24); printf("Is printer ready (Y/N) ? ");
     gets(B); B[0]=toupper(B[0]);
     if(B[0]=='Y') prt_policy();
   };
   modules();
   return;
 }

 void solidwas()
 { /* SOLID WASTE CHARACTERISTICS */
   int c,i,n,h;
   rockford();
   fil = fopen(D[MODULE][INDEX],"r");
   if ( fil == 0 )
   {
     printf("Error          opening          file          |%s|          %d
%d",D[MODULE][INDEX],MODULE,INDEX);
     sleep(DELAY);
     exit(0);
   };
   for(i=1;i<20;i++)
   {
     W[i]=0.0;
     fgets(B,MFL,fil);
     sscanf(B,"%f",&W[i]);
   };
   clrscr();
   frame_screen();
   show_default_file(D[MODULE][INDEX]);
   /* this module uses 2 screens */
   gotoxy(20,1);
   printf(" Solid Waste Characteristics (screen 1) ");
   gotoxy(12,3);
   printf("(Most values in this module are universal and will");
   gotoxy(12,4);
   printf("not vary across different collection system types.)");
   gotoxy(5,6);
   printf("Average amount of solid waste created ");
   gotoxy(5,7);
   printf("per  person  per  month  (lbs/month).....
%12.2f",W[1]);
   gotoxy(5,9);
   printf("Percentage (0-100) of soid waste that is:");
   gotoxy(20,11);
   printf("Aluminum..... %12.2f",W[2]);
   gotoxy(20,12);
   printf("Newspaper..... %12.2f",W[3]);
   gotoxy(20,13);
```

```
    printf("Glass..... %12.2f",W[4]);
    gotoxy(20,14);
    printf("Plastics..... %12.2f",W[5]);
    gotoxy(20,15);
    printf("Other material of interest..... %12.2f",W[6]);
    gotoxy(5,17);
    printf("Percentage (0-100) of plastics that are:");
    gotoxy(20,19);
    printf("PE..... %12.2f",W[7]);
    gotoxy(20,20);
    printf("PET..... %12.2f",W[8]);
    XX[1]=XX[2]=XX[3]=XX[4]=XX[5]=XX[6]=XX[7]=XX[8]=XX[9]=XX[10]=XX[11]=65;
    YY[1]=7;YY[2]=11;YY[3]=12;YY[4]=13;YY[5]=14;YY[6]=15;YY[7]=19;YY[8]=20;
    n=0;
    for(;;)
    {
      n++;
      if(n>8)break;
      gotoxy( XX[n],YY[n]);
      h = get_key_2();
      if ( h == -27 ) break;
      else if ( h == 0 ) W[n] = v ;
      gotoxy( XX[n],YY[n]);
      printf("            ");
      gotoxy( XX[n],YY[n]);
      printf("%12.2f",W[n]);
      if(h==-85)
        if(n==1)n=0;
        else n-=2;
    };
    sleep(DELAY);
    clrscr();
    frame_screen();
    show_default_file(D[MODULE][INDEX]);
    gotoxy(20,1);
    printf(" Solid Waste Characteristics (screen 2) ");
    gotoxy(5,3);
    printf("Percentage (0-100) of PET that is:");
    gotoxy(20,5);
    printf("Clear..... %12.2f",W[9]);
    gotoxy(20,6);
    printf("Colored..... %12.2f",W[10]);
    gotoxy(5,8);
    printf("Percentage (0-100) of glass that is:");
    gotoxy(20,10);
    printf("Clear..... %12.2f",W[11]);
    gotoxy(20,11);
    printf("Green..... %12.2f",W[12]);
    gotoxy(20,12);
    printf("Brown..... %12.2f",W[13]);
    gotoxy(5,14);
    printf("Average density (lb/ft^3) of solid waste.....
%12.2f",W[14]);
    gotoxy(5,16);
```

```
printf("Average density (lb/ft^3) of recyclable:");
gotoxy(20,18);
printf("Aluminum..... %12.2f",W[15]);
gotoxy(20,19);
printf("Newspaper..... %12.2f",W[16]);
gotoxy(20,20);
printf("Glass..... %12.2f",W[17]);
gotoxy(20,21);
printf("Plastic..... %12.2f",W[18]);
gotoxy(20,22);
printf("Other Material..... %12.2f",W[19]);
YY[1]=5;YY[2]=6;YY[3]=10;YY[4]=11;YY[5]=12;YY[6]=14;YY[7]=18;
YY[8]=19;YY[9]=20;YY[10]=21;YY[11]=22;
n=0;
for(;;)
{
  n++;
  if(n>11)break;
  gotoxy( XX[n],YY[n]);
  h = get_key_2();
  if ( h == -27 ) break;
  else if ( h == 0 ) W[n+8] = v ;
  gotoxy( XX[n],YY[n]);
  printf("          ");
  gotoxy( XX[n],YY[n]);
  printf("%12.2f",W[n+8]);
  if(h==-85)
    if(n==1)n=0;
    else n-=2;
};
gotoxy(5,24); printf("                          ");
gotoxy(5,24); printf("Update changes (Y/N) ? ");
c = get_key_1();
if ( c == -89 )
{ /* Save the record */
  fil = fopen(D[MODULE][INDEX],"w");
  if ( fil == 0 ) flag_unopened_file(D[MODULE][INDEX]);
  for(i=1;i<20;i++) fprintf(fil,"%f\n",W[i]);
  save_current_file();
}
else
{
  if ( NEW_INPUT_FILE == 1) unlink( D[MODULE][INDEX] ) ;
  main_menu();
};
gotoxy(5,24); printf("                                ");
gotoxy(5,24); printf("Print Hardcopy (Y/N) ? ");
gets(B);
B[0]=toupper(B[0]);
if(B[0]=='Y')   /*Do not save but print the data */
{
  gotoxy(5,24); printf("                                ");
  gotoxy(5,24); printf("Is printer ready (Y/N) ? ");
  gets(B); B[0]=toupper(B[0]);
```

```
    if(B[0]=='Y') prt_solidwas();
  };
  modules();
  return;
}

void collect()
{/* #4 COLLECTION EQUIPMENT CHARACTERISTICS */
  int c,i,j,k,h;
  rockford();
  fil = fopen(D[MODULE][INDEX],"r");
  if ( fil == 0 ) flag_unopened_file(D[MODULE][INDEX]);
  for(i=1;i<33;i++)
  {
    C[i]=0.0;
    fgets(B,MFL,fil);
    sscanf(B,"%f",&C[i]);
  };
  fclose(fil);
  clrscr();
  frame_screen();
  show_default_file(D[MODULE][INDEX]);
  gotoxy(30,1);printf("Collection Equipment");
  gotoxy(35,3);printf("Vehicle Type");
  gotoxy(24,4);printf("          1                2                3
4   ");
  gotoxy(5,5);printf("-----------------------------------------------");
  gotoxy(2,6);printf("# of Units");
  for(i=1;i<26;i+=8)
  {
    gotoxy((24+ i*14/8 ),6);
    printf("%12.2f",C[i]);
  };
  gotoxy(2,8);printf("Weight Capacity (lbs)");
  for(i=2;i<27;i+=8)
  {
    gotoxy((24+(i-1)*14/8),8);
    printf("%12.2f",C[i]);
  };
  gotoxy(2,10);printf("Volume Capacity (ft^3)");
  for(i=3;i<28;i+=8)
  {
    gotoxy((24+(i-2)*14/8),10);
    printf("%12.2f",C[i]);
  };
  gotoxy(2,12);printf("Equipment Cost");
  gotoxy(2,13);printf("/month/unit");
  for(i=4;i<29;i+=8)
  {
    gotoxy((24+(i-3)*14/8),12);
    printf("%12.2f",C[i]);
  };
  gotoxy(2,15);printf("Overhead");
  gotoxy(2,16);printf("/month/unit");
```

```
for(i=5;i<30;i+=8)
{
  gotoxy((24+(i-4)*14/8),15);
  printf("%12.2f",C[i]);
};
gotoxy(2,18);printf("Cost/mile/unit");
for(i=6;i<31;i+=8)
{
  gotoxy((24+(i-5)*14/8),18);
  printf("%12.2f",C[i]);
};
gotoxy(2,20);printf("Staff/unit (frac ok)");
for(i=7;i<32;i+=8)
{
  gotoxy((24+(i-6)*14/8),20);
  printf("%12.2f",C[i]);
};
gotoxy(2,22);printf("Avg # Collect");
gotoxy(2,23);printf("points/month/unit");
for(i=8;i<33;i+=8)
{
  gotoxy((24+(i-7)*14/8),22);
  printf("%12.2f",C[i]);
};
XX[1]=25;XX[2]=39;XX[3]=53;XX[4]=67;
YY[1]=6;YY[2]=8;YY[3]=10;YY[4]=12;YY[5]=15;YY[6]=18;YY[7]=20;YY[8]=22;
k = 0;
j = 1;
for(;;)
{
  k++;
  if(k>8)
  {
    k = 1 ;
    j++;
  };
  if(j>4)break;
  i=(j<<3)+k-8;
  gotoxy(XX[j],YY[k]);
  h=get_key_2();
  if(h == -27 ) break;
  else if ( h == 0 ) C[i]=v;
  gotoxy(XX[j],YY[k]);
  printf("           ");
  gotoxy(XX[j],YY[k]);
  printf("%12.2f",C[i]);
  if (h == -85)
     if (k == 1)
     {
       if ( j > 1 )
       {
         k=7;
         j--;
       }
```

```
          else j=1;
        }
        else k-=2;
    };
    cik:;
    gotoxy(5,24); printf("                          ");
    gotoxy(5,24); printf("Update changes (Y/N) ? ");
    c = get_key_1();
    if ( c == -89 )
    { /* Save the record */
      fil = fopen(D[MODULE][INDEX],"w");
      if ( fil == 0 ) flag_unopened_file(D[MODULE][INDEX]);
      for(i=1;i<33;i++) fprintf(fil,"%f\n",C[i]);
      save_current_file();
    }
    else
    {
      if ( NEW_INPUT_FILE == 1) unlink( D[MODULE][INDEX] ) ;
      main_menu();
    };
    gotoxy(5,24); printf("                                ");
    gotoxy(5,24); printf("Print Hardcopy (Y/N) ? ");
    gets(B);
    B[0]=toupper(B[0]);
    if(B[0]=='Y')   /*Do not save but print the data */
    {
      gotoxy(5,24); printf("                                ");
      gotoxy(5,24); printf("Is printer ready (Y/N) ? ");
      gets(B); B[0]=toupper(B[0]);
      if(B[0]=='Y') prt_collect();
    };
    modules();
    return;
}

void labor()
{/* #5 LABOR COSTS */
    int c,i,n,h;
    rockford();
    fil = fopen(D[MODULE][INDEX],"r");
    if ( fil == 0 ) flag_unopened_file(D[MODULE][INDEX]);
    for(i=1;i<3;++i)
    {
      L[i]=0.0;
      fgets(B,MFL,fil);
      sscanf(B,"%f",&L[i]);
    };
    fclose(fil);
    clrscr();
    frame_screen();
    show_default_file(D[MODULE][INDEX]);
    gotoxy(35,1);printf("Labor Costs");
    gotoxy(5,4);printf("The average monthly cost per laborer for:");
    gotoxy(20,6);printf("Collectors.............%12.2f",L[1]);
```

```
    gotoxy(20,8);printf("Material Handlers......%12.2f",L[2]);
    XX[1]=XX[2]=43;
    YY[1]=6;YY[2]=8;
    n=0;
    for(;;)
    {
      n++;
      if (n == 3)  break;
      gotoxy(XX[n],YY[n]);
      h=get_key_2();
      if (h == -27) break;
      else if (h==0) L[n]=v;
      gotoxy(XX[n],YY[n]);
      printf("            ");
      gotoxy(XX[n],YY[n]);
      printf("%12.2f",L[n]);
      if (h == -85)
         if (n == 1) n=0;
         else n-=2;
    };
    gotoxy(5,15); printf("                          ");
    gotoxy(5,15); printf("Update changes (Y/N) ? ");
    c = get_key_1();
    if ( c == -89 )
    { /* Save the record */
      fil = fopen(D[MODULE][INDEX],"w");
      if ( fil == 0 ) flag_unopened_file(D[MODULE][INDEX]);
      for(i=1;i<3;i++) fprintf(fil,"%f\n",L[i]);
      save_current_file();
    }
    else
    {
      if ( NEW_INPUT_FILE == 1) unlink( D[MODULE][INDEX] ) ;
      main_menu();
    };
    gotoxy(5,15); printf("                              ");
    gotoxy(5,15); printf("Print Hardcopy (Y/N) ?  ");
    gets(B);
    B[0]=toupper(B[0]);
    if(B[0]=='Y')   /*Do not save but print the data */
    {
      gotoxy(5,15); printf("                              ");
      gotoxy(5,15); printf("Is printer ready (Y/N) ? ");
      gets(B); B[0]=toupper(B[0]);
      if(B[0]=='Y') prt_labor();
    };
    modules();
    return;
}

void material()
{/* #6 MATERIAL HANDLING */
    int c,i,n,h;
    rockford();
```

```
   fil = fopen(D[MODULE][INDEX],"r");
   if ( fil == 0 ) flag_unopened_file(D[MODULE][INDEX]);
   for(i=1;i<12;i++)
   {
     H[i]=0.0;
     fgets(B,MFL,fil);
     sscanf(B,"%f",&H[i]);
   };
   fclose(fil);
   clrscr();
   frame_screen();
   show_default_file(D[MODULE][INDEX]);
   gotoxy(32,1);printf(" Material Handling ");
   gotoxy(23,4);printf("                  # operators    # machines     equip
cost");
   gotoxy(10,6);printf("-------------------------------------------------");
   gotoxy(62,5);printf("($/month/unit)");
   gotoxy(10,7);printf("Sorting-manual");
   gotoxy(37,7);printf("%12.2f",H[1]);
   gotoxy(60,7);printf("n/a");
   gotoxy(74,7);printf("n/a");
   for(i=1;i<4;i++)
   {
     gotoxy((23+i*14),9);
     printf("%12.2f",H[i+1]);
   };
   gotoxy(10,9);printf("Granulation/Shredding");
   for(i=1;i<4;i++)
   {
     gotoxy((23+i*14),11);
     printf("%12.2f",H[i+4]);
   };
   gotoxy(10,11);printf("Baling");
   for(i=1;i<4;i++)
   {
     gotoxy((23+i*14),13);
     printf("%12.2f",H[i+7]);
   };
   gotoxy(10,13);printf("Crushing (glass/cans only)");
   gotoxy(5,17);
   printf("Monthly     overhead     for     the     material     handling
facility....%12.2f",H[11]);
   XX[1]=37;XX[2]=37;XX[3]=51;XX[4]=65;XX[5]=37;XX[6]=51;
   XX[7]=65;XX[8]=37;XX[9]=51;XX[10]=65;XX[11]=60;
   YY[1]=7;YY[2]=9;YY[3]=9;YY[4]=9;YY[5]=11;YY[6]=11;YY[7]=11;
   YY[8]=13;YY[9]=13;YY[10]=13;YY[11]=17;
   n=0;
   for(;;)
   {
     n++;
     if (n == 12)  break;
     gotoxy(XX[n],YY[n]);
     h=get_key_2();
     if (h == -27) break;
```

```
    else if (h==0) H[n]=v;
    gotoxy(XX[n],YY[n]);
    printf("          ");
    gotoxy(XX[n],YY[n]);
    printf("%12.2f",H[n]);
    if (h == -85)
       if (n == 1) n=0;
       else n-=2;
  };
  gotoxy(5,19); printf("                          ");
  gotoxy(5,19); printf("Update changes (Y/N) ? ");
  c = get_key_1();
  if ( c == -89 )
  { /* Save the record */
    fil = fopen(D[MODULE][INDEX],"w");
    if ( fil == 0 ) flag_unopened_file(D[MODULE][INDEX]);
    for(i=1;i<12;i++) fprintf(fil,"%f\n",H[i]);
    save_current_file();
  }
  else
  {
    if ( NEW_INPUT_FILE == 1) unlink( D[MODULE][INDEX] ) ;
    main_menu();
  };
  gotoxy(5,19); printf("                              ");
  gotoxy(5,19); printf("Print Hardcopy (Y/N) ? ");
  gets(B);
  B[0]=toupper(B[0]);
  if(B[0]=='Y')   /*Do not save but print the data */
  {
    gotoxy(5,19); printf("                              ");
    gotoxy(5,19); printf("Is printer ready (Y/N) ? ");
    gets(B); B[0]=toupper(B[0]);
    if(B[0]=='Y') prt_material();
  };
  modules();
  return;
}

void shipping()
{/* #7  SHIPPING COSTS */
  int c,i,n,h;
  rockford();
  fil=fopen(D[MODULE][INDEX],"r");
  if ( fil == 0 ) flag_unopened_file(D[MODULE][INDEX]);
  for(i=1;i<19;i++)
  {
    S[i]=0.0;
    fgets(B,MFL,fil);
    sscanf(B,"%f",&S[i]);
  };
  fclose(fil);
  clrscr();
  frame_screen();
```

```
show_default_file(D[MODULE][INDEX]);
gotoxy(33,1);printf(" Shipping Costs (screen 1) ");
gotoxy(5,3);
printf("Distance (miles) from the materials handling facility to:");
gotoxy(20,5); printf("Glass market..... %12.2f",S[1]);
gotoxy(20,6); printf("Newspaper market..... %12.2f",S[2]);
gotoxy(20,7); printf("Aluminum market..... %12.2f",S[3]);
gotoxy(20,8); printf("Plastic processing plant..... %12.2f",S[4]);
gotoxy(20,9); printf("Landfill..... %12.2f",S[5]);
gotoxy(20,10);printf("Market for other materials... %12.2f",S[6]);
gotoxy(5,12); printf("Cost ($) per mile per shipment for:");
gotoxy(20,14);printf("Glass..... %12.2f",S[7]);
gotoxy(20,15);printf("Newspaper..... %12.2f",S[8]);
gotoxy(20,16);printf("Aluminum..... %12.2f",S[9]);
gotoxy(20,17);printf("Plastic..... %12.2f",S[10]);
gotoxy(20,18);printf("Solid waste..... %12.2f",S[11]);
gotoxy(20,19);printf("Other material..... %12.2f",S[12]);
XX[1]=XX[2]=XX[3]=XX[4]=XX[5]=XX[6]=50;
XX[7]=XX[8]=XX[9]=XX[10]=XX[11]=XX[12]=50;
YY[1]=5;YY[2]=6;YY[3]=7;YY[4]=8;YY[5]=9;YY[6]=10;
YY[7]=14;YY[8]=15;YY[9]=16;YY[10]=17;YY[11]=18;YY[12]=19;
n=0;
for(;;)
{
  n++;
  if(n> 12)break;
  gotoxy(XX[n],YY[n]);
  h=get_key_2();
  if (h == -27) break;
  else if (h==0) S[n]=v;
  gotoxy(XX[n],YY[n]);
  printf("          ");
  gotoxy(XX[n],YY[n]);
  printf("%12.2f",S[n]);
  if(h==-85)
    if(n==1)n=0;
    else n-=2;
};
sleep(DELAY);
clrscr();
frame_screen();
show_default_file(D[MODULE][INDEX]);
gotoxy(33,1);printf(" Shipping Costs (screen 2) ");
gotoxy(5,3);printf("Weight capacity (lbs) per shipment of:");
gotoxy(20,5); printf("Glass..... %12.2f",S[13]);
gotoxy(20,6); printf("Newspaper..... %12.2f",S[14]);
gotoxy(20,7); printf("Aluminum..... %12.2f",S[15]);
gotoxy(20,8); printf("Plastic..... %12.2f",S[16]);
gotoxy(20,9); printf("Solid waste..... %12.2f",S[17]);
gotoxy(20,10);printf("Other material... %12.2f",S[18]);
XX[1]=XX[2]=XX[3]=XX[4]=XX[5]=XX[6]=38;
YY[1]=5;YY[2]=6;YY[3]=7;YY[4]=8;YY[5]=9;YY[6]=10;
n=12;
for(;;)
```

```
    {
      n++;
      if(n>18)break;
      gotoxy(XX[n-12],YY[n-12]);
      h=get_key_2();
      if (h == -27) break;
      else if (h==0) S[n]=v;
      gotoxy(XX[n-12],YY[n-12]);
      printf("           ");
      gotoxy(XX[n-12],YY[n-12]);
      printf("%12.2f",S[n]);
      if(h==-85)
        if(n==13)n=12;
        else n-=2;
    };
    gotoxy(5,15); printf("                          ");
    gotoxy(5,15); printf("Update changes (Y/N) ? ");
    c = get_key_1();
    if ( c == -89 )
    { /* Save the record */
      fil = fopen(D[MODULE][INDEX],"w");
      if ( fil == 0 ) flag_unopened_file(D[MODULE][INDEX]);
      for(i=1;i<19;i++) fprintf(fil,"%f\n",S[i]);
      save_current_file();
    }
    else
    {
      if ( NEW_INPUT_FILE == 1) unlink( D[MODULE][INDEX] ) ;
      main_menu();
    };
    gotoxy(5,15); printf("                              ");
    gotoxy(5,15); printf("Print Hardcopy (Y/N) ?  ");
    gets(B);
    B[0]=toupper(B[0]);
    if(B[0]=='Y')   /*Do not save but print the data */
    {
      gotoxy(5,15); printf("                          ");
      gotoxy(5,15); printf("Is printer ready (Y/N) ? ");
      gets(B); B[0]=toupper(B[0]);
      if(B[0]=='Y') prt_shipping();
    };
    modules();
    return;
}

void disposal()
{/* #8 DISPOSAL COSTS */
    int c,i,n,h;
    rockford();
    fil = fopen(D[MODULE][INDEX],"r");
    if ( fil == 0 ) flag_unopened_file(D[MODULE][INDEX]);
    for(i=1;i<3;i++)
    {
      DIS[i]=0.0;
```

```
    fgets(B,MFL,fil);
    sscanf(B,"%f",&DIS[i]);
  }
  fclose(fil);
  clrscr();
  frame_screen();
  show_default_file(D[MODULE][INDEX]);
  gotoxy(34,1);printf("Disposal Costs");
  gotoxy(5,4);printf("Tipping      fee      ($/ton).....
%12.2f",DIS[1]);
  gotoxy(5,6);printf("Government   subsidy   or   abatement   ($/ton)..
%12.2f",DIS[2]);
  XX[1]=XX[2]=47;
  YY[1]=4;YY[2]=6;
  n=0;
  for(;;)
  {
    n++;
    if (n == 3)  break;
    gotoxy(XX[n],YY[n]);
    h=get_key_2();
    if (h==-27) break;
    else if (h==0) DIS[n]=v;
    gotoxy(XX[n],YY[n]);
    printf("          ");
    gotoxy(XX[n],YY[n]);
    printf("%12.2f",DIS[n]);
    if (h ==-85)
       if (n == 1) n=0;
       else n-=2;
  };
  gotoxy(5,15); printf("                          ");
  gotoxy(5,15); printf("Update changes (Y/N) ? ");
  c = get_key_1();
  if ( c == -89 )
  { /* Save the record */
    fil = fopen(D[MODULE][INDEX],"w");
    if ( fil == 0 ) flag_unopened_file(D[MODULE][INDEX]);
    for(i=1;i<3;i++) fprintf(fil,"%f\n",DIS[i]);
    save_current_file();
  }
  else
  {
    if ( NEW_INPUT_FILE == 1) unlink( D[MODULE][INDEX] ) ;
    main_menu();
  };
  gotoxy(5,15); printf("                            ");
  gotoxy(5,15); printf("Print Hardcopy (Y/N) ?  ");
  gets(B);
  B[0]=toupper(B[0]);
  if(B[0]=='Y')   /*Do not save but print the data */
  {
    gotoxy(5,15); printf("                          ");
    gotoxy(5,15); printf("Is printer ready (Y/N) ? ");
```

```
    gets(B); B[0]=toupper(B[0]);
    if(B[0]=='Y') prt_disposal();
  };
  modules();
  return;
}

void plasproc()
{ /* #9 PLASTICS PROCESSING COSTS */
  int c,i,n,h;
  rockford();
  fil = fopen(D[MODULE][INDEX],"r");
  if ( fil == 0 ) flag_unopened_file(D[MODULE][INDEX]);
  for(i=1;i<6;i++)
  {
    PP[i]=0.0;
    fgets(B,MFL,fil);
    sscanf(B,"%f",&PP[i]);
  };
  fclose(fil);
  clrscr();
  frame_screen();
  show_default_file(D[MODULE][INDEX]);
  gotoxy(26,1);printf(" Plastics Processing Costs ");
  gotoxy(10,4);printf("Equipment              cost/month.....
%12.2f",PP[1]);
  gotoxy(10,6);printf("Direct     process     costs     ($/ton).....
%12.2f",PP[2]);
  gotoxy(10,8);printf("Processing     facility     monthly     overhead..
%12.2f",PP[3]);
  gotoxy(10,10);printf("#    of    operators    at    processing    plant....
%12.2f",PP[4]);
  gotoxy(10,12);printf("Process     efficiency     ratio     (0-1).....
%12.2f",PP[5]);
  XX[1]=XX[2]=XX[3]=XX[4]=XX[5]=49;
  YY[1]=4;YY[2]=6;YY[3]=8;YY[4]=10;YY[5]=12;
  n=0;
  for(;;)
  {
    n++;
    if (n == 6)  break;
    gotoxy(XX[n],YY[n]);
    h=get_key_2();
    if (h==-27) break;
    else if (h==0) PP[n]=v;
    gotoxy(XX[n],YY[n]);
    printf("           ");
    gotoxy(XX[n],YY[n]);
    printf("%12.2f",PP[n]);
    if (h == -85)
       if (n ==1) n=0;
       else n-=2;
 };
 gotoxy(5,20); printf("                          ");
```

```
  gotoxy(5,20); printf("Update changes (Y/N) ? ");
  c = get_key_1();
  if ( c == -89 )
  { /* Save the record */
    fil = fopen(D[MODULE][INDEX],"w");
    if ( fil == 0 ) flag_unopened_file(D[MODULE][INDEX]);
    for(i=1;i<6;i++) fprintf(fil,"%f\n",PP[i]);
    save_current_file();
  }
  else
  {
    if ( NEW_INPUT_FILE == 1) unlink( D[MODULE][INDEX] ) ;
    main_menu();
  };
  gotoxy(5,20); printf("                                ");
  gotoxy(5,20); printf("Print Hardcopy (Y/N) ?  ");
  gets(B);
  B[0]=toupper(B[0]);
  if(B[0]=='Y')   /*Do not save but print the data */
  {
    gotoxy(5,20); printf("                                ");
    gotoxy(5,20); printf("Is printer ready (Y/N) ? ");
    gets(B); B[0]=toupper(B[0]);
    if(B[0]=='Y') prt_plasproc();
  };
  modules();
  return;
}

void selling()
{/* =10 SELLING PRICES OF RECYCLED WASTE MATERIAL  */
  int c,i,n,h;
  rockford();
  fil = fopen(D[MODULE][INDEX],"r");
  if ( fil == 0 ) flag_unopened_file(D[MODULE][INDEX]);
  for(i=1;i<14;i++)
  {
    PR[i]=0.0;
    fgets(B,MFL,fil);
    sscanf(B,"%f",&PR[i]);
  };
  fclose(fil);
  clrscr();
  frame_screen();
  show_default_file(D[MODULE][INDEX]);
  gotoxy(19,1);printf(" Selling Prices of Recycled Waste Material ");
  gotoxy(10,3);printf("Price per lb of:");
  gotoxy(20,5);printf("PE..... %12.2f",PR[1]);
  gotoxy(20,6);printf("Clear PET..... %12.2f",PR[2]);
  gotoxy(20,7);printf("Colored PET..... %12.2f",PR[3]);
  gotoxy(20,8);printf("Clear glass..... %12.2f",PR[4]);
  gotoxy(20,9);printf("Green glass..... %12.2f",PR[5]);
  gotoxy(20,10);printf("Brown glass..... %12.2f",PR[6]);
  gotoxy(20,11);printf("Aluminum..... %12.2f",PR[7]);
```

```
  gotoxy(20,12);printf("Newspaper..... %12.2f",PR[8]);
  gotoxy(20,13);printf("Mixed PE/PET..... %12.2f",PR[9]);
  gotoxy(20,14);printf("Mixed glass..... %12.2f",PR[10]);
  gotoxy(20,15);printf("Other material of interest.. %12.2f",PR[11]);
  gotoxy(5,17);
  printf("Are you selling (0)-mixed or (1)-sorted PE/PET...
%6.0f",PR[12]);
  gotoxy(5,18);
  printf("Are you selling (0)-mixed or (1)-sorted glass....
%6.0f",PR[13]);
  XX[1]=XX[2]=XX[3]=XX[4]=XX[5]=XX[6]=49;
  XX[7]=XX[8]=XX[9]=XX[10]=XX[11]=49;XX[12]=XX[13]=55;
  YY[1]=5;YY[2]=6;YY[3]=7;YY[4]=8;YY[5]=9;YY[6]=10;
  YY[7]=11;YY[8]=12;YY[9]=13;YY[10]=14;YY[11]=15;YY[12]=17;YY[13]=18;
  n=0;
  for(;;)
  {
    n++;
    if (n == 14)  break;
    gotoxy(XX[n],YY[n]);
    h=get_key_2();
    if (h == -27) break;
    else if (h==0) PR[n]=v;
    gotoxy(XX[n],YY[n]);
    printf("           ");
    gotoxy(XX[n],YY[n]);
    if(n == 12 ) printf("%6.0f",PR[n]);
    else if ( n == 13 )printf("%6.0f",PR[n]);
    else printf("%12.2f",PR[n]);
    if (h == -85)
       if (n == 1) n=0;
       else n-=2;
    if ( n == 12 ) if ( PR[12] != 1.0 ) PR[12] =0.0 ;
    if ( n == 13 ) if ( PR[13] != 1.0 ) PR[13] =0.0 ;
  };
  gotoxy(5,21); printf("                          ");
  gotoxy(5,21); printf("Update changes (Y/N) ? ");
  c = get_key_1();
  if ( c == -89 )
  { /* Save the record */
    fil = fopen(D[MODULE][INDEX],"w");
    if ( fil == 0 ) flag_unopened_file(D[MODULE][INDEX]);
    for(i=1;i<14;i++) fprintf(fil,"%f\n",PR[i]);
    save_current_file();
  }
  else
  {
    if ( NEW_INPUT_FILE == 1) unlink( D[MODULE][INDEX] ) ;
    main_menu();
  };
  gotoxy(5,21); printf("                              ");
  gotoxy(5,21); printf("Print Hardcopy (Y/N) ?  ");
  gets(B);
  B[0]=toupper(B[0]);
```

```
  if(B[0]=='Y')   /*Do not save but print the data */
  {
    gotoxy(5,21); printf("                              ");
    gotoxy(5,21); printf("Is printer ready (Y/N) ? ");
    gets(B); B[0]=toupper(B[0]);
    if(B[0]=='Y') prt_selling();
  };
  modules();
  return;
}

void colpoint()
{/*  #11 COLLECTION POINT DATA  */
  int c,i,n,h;
  rockford();
  fil = fopen(D[MODULE][INDEX],"r");
  if ( fil == 0 ) flag_unopened_file(D[MODULE][INDEX]);
  for(i=1;i<3;i++)
  {
    PT[i]=0.0;
    fgets(B,MFL,fil);
    sscanf(B,"%f",&PT[i]);
  };
  fclose(fil);
  clrscr();
  frame_screen();
  show_default_file(D[MODULE][INDEX]);
  gotoxy(30,1);printf(" Collection Point Data ");
  gotoxy(5,4);printf("Equipment    cost    per    collection    point..
%12.2f",PT[1]);
  gotoxy(5,6);printf("Overhead     per     collection     point.......
%12.2f",PT[2]);
  XX[1]=XX[2]=43;
  YY[1]=4;YY[2]=6;
  n=0;
  for(;;)
  {
    n++;
    if(n== 3) break;
    gotoxy(XX[n],YY[n]);
    h=get_key_2();
    if (h==-27) break;
    else if (h==0) PT[n]=v;
    gotoxy(XX[n],YY[n]);
    printf("           ");
    gotoxy(XX[n],YY[n]);
    printf("%12.2f",PT[n]);
    if (h == -85)
       if(n == 1) n=0;
       else n-=2;
  };
  gotoxy(5,15); printf("                          ");
  gotoxy(5,15); printf("Update changes (Y/N) ? ");
  c = get_key_1();
```

```
  if ( c == -89 )
  { /* Save the record */
    fil = fopen(D[MODULE][INDEX],"w");
    if ( fil == 0 ) flag_unopened_file(D[MODULE][INDEX]);
    for(i=1;i<3;i++) fprintf(fil,"%f\n",PT[i]);
    save_current_file();
  }
  else
  {
    if ( NEW_INPUT_FILE == 1) unlink( D[MODULE][INDEX] ) ;
    main_menu();
  }
  gotoxy(5,15); printf("                                ");
  gotoxy(5,15); printf("Print Hardcopy (Y/N) ? ");
  gets(B);
  B[0]=toupper(B[0]);
  if(B[0]=='Y')   /*Do not save but print the data */
  {
    gotoxy(5,15); printf("                              ");
    gotoxy(5,15); printf("Is printer ready (Y/N) ? ");
    gets(B); B[0]=toupper(B[0]);
    if(B[0]=='Y') prt_colpoint();
  };
  modules();
  return;
}
```

```
/* compute.c */

#include "variable.h"

float computations()
{/* CALCULATES ALL VARIABLES */
 /* Provisions are made to keep from dividing by zero */
 /* RAW MATERIAL */
  int c,i;
  float EPSILON = 0.0001;
  RM[1]=W[1]*R[2]*R[1];
  RM[2]=RM[1]*P[1]*W[2]/10000;
  RM[3]=RM[1]*P[2]*W[3]/10000;
  RM[4]=RM[1]*P[3]*W[4]/10000;
  RM[5]=RM[1]*P[4]*W[5]/10000;
  RM[6]=RM[1]*P[5]*W[6]/10000;
  RM[7]=RM[5]*W[7]/100;
  RM[8]=RM[5]*W[8]/100;
  RM[9]=RM[8]*W[9]/100;
  RM[10]=RM[8]*W[10]/100;
  RM[11]=RM[4]*W[11]/100;
  RM[12]=RM[4]*W[12]/100;
  RM[13]=RM[4]*W[13]/100;
/* check for zero densities */
  cdbyzero();
  RM[14]=W[14]*VOLTOT*27;
  RM[15]=RM[2]+RM[3]+RM[4]+RM[5]+RM[6];
  RM[16]=RM[1]-RM[15];
  for(i=1;i<17;i++) if ( abs( RM[i] ) < EPSILON ) RM[i]=0.0;
  /*COLLECTION POINT */
  CP[1]=RM[5]*P[6];
  CP[2]=PT[1]*R[4];
  CP[3]=PT[2]*R[4];
  CP[4]=CP[1]+P[7]+CP[2]+CP[3];
  for(i=1;i<5;i++) if ( abs( CP[i] ) < EPSILON ) CP[i]=0.0;
  /*COLLECTION SYSTEM */
  if (R[4]==0.0)
  {
    printf("For this analysis you can not have zero collection points.");
    printf("\x7 R[4] == 0");
    wait_and_go();
  } ;
  CS[1]=R[3]/R[4];
  CS[2]=L[1]*(C[1]*C[7]+C[9]*C[15]+C[17]*C[23]+C[25]*C[31]);
  CS[3]=C[4]*C[1]+C[12]*C[9]+C[20]*C[17]+C[28]*C[25];
  CS[4]=C[5]*C[1]+C[13]*C[9]+C[21]*C[17]+C[29]*C[25];
  CS[5]=CS[1]*(C[8]*C[1]*C[6]+C[16]*C[9]*C[14]+C[24]
        *C[17]*C[22]+C[32]*C[25]*C[30]);
  CS[6]=CS[2]+CS[3]+CS[4]+CS[5];
    for(i=1;i<7;i++) if ( abs( CS[i] ) < EPSILON ) CS[i]=0.0;
  COPO=C[8]*C[1]+C[16]*C[9]+C[24]*C[17]+C[32]*C[25];
  if ((COPO < .95*R[4]*P[8]) || (COPO > 1.05*R[4]*P[8]))
  {
    clrscr();
```

```
    gotoxy(5,20);printf("COPO is %f r4 is %f p8 is %f\n",COPO,R[4],P[8]);
    gotoxy(5,1);printf("The total number of collection points handled by
");
    gotoxy(5,2);printf("collection equipment is inconsistent (greater
than ");
    gotoxy(5,3);printf("5%% error) with the total number of collection
points");
    gotoxy(5,4);printf("input in the Regional Data module.");
    gotoxy(30,10);printf("(1)  Continue");
    gotoxy(30,12);printf("(2)  Return to main menu");
    c=get_key_1();
    if(c == 2 )
    {
      clrscr();
      main_menu();
    }
    clrscr();
    printf("Running...");
  }
 /* MATERIALS HANDLING */
  MH[1]=H[1]+H[2]+H[5]+H[8];
  MH[2]=L[2]*MH[1];
  MH[3]=H[3]*H[4]+H[6]*H[7]+H[9]*H[10];
  MH[4]=MH[3]+MH[2]+H[11];
  for(i=1;i<5;i++) if ( abs( MH[i] ) < EPSILON ) MH[i]=0.0;
  /* SHIPPING */
  if(RM[2]==0.0) SH[1]=0.0;
  else if(S[15]==0.0)
  {
    printf("Weight capacity per shipement of aluminum can not be equal
zero\n");
    printf("\x7 S[15]==0");
    wait_and_go();
  }
  else SH[1]=RM[2]*S[3]*S[9]/S[15];
  if(RM[4]==0.0) SH[2] = 0.0;
  else
  {
    if(S[13]==0.0)
    {
      printf("Weight capacity per shipment of glass can not equal
zero.\n");
      printf("\x7 S[13]==0");
      wait_and_go();
    }
    else SH[2]=RM[4]*S[1]*S[7]/S[13];
  }
  if( RM[3] == 0.0) SH[3] = 0.0;
  else
  {
    if( S[14] == 0.0)
    {
      printf("Weight capacity per shipment of newspaper can not equal
zero.\n");
```

```
      printf("\x7 S[14]==0");
      wait_and_go();
    }
    else SH[3]=RM[3]*S[2]*S[8]/S[14];
  }
  if( RM[5] == 0.0) SH[4] = 0.0;
  else
  {
    if(S[16]==0.0)
    {
      printf("Weight capacity per shipment of plastic can not equal
zero.\n");
      printf("\x7 S[16]==0");
      wait_and_go();
    }
    else SH[4]=RM[3]*S[4]*S[10]/S[16];
  }
  if( RM[6] == 0.0 ) SH[5]=0.0;
  else
  {
    if(S[18]==0.0)
    {
      printf("Weight capacity per shipment of Other Material can not
equal zero.\n");
      printf("\x7 s[18]==0");
      wait_and_go();
    }
    else SH[5]=RM[6]*S[6]*S[12]/S[18];
  }
  if( RM[16] == 0.0 ) SH[6] = 0.0;
  else
  {
    if( P[9] == 0.0 ) S[17] = 1 ;
    else
    {
      if( S[17] == 0 )
      {
        printf("Weight capacity per shipment of Solid Waste can not equal
zero.\n");
        printf("\x7 S[17]==0");
        wait_and_go();
      }
    }
    S[6] = RM[16]*S[6]*S[11]/S[17];
  }
  SH[7]=SH[1]+SH[2]+SH[3]+SH[4]+SH[5];
  for(i=1;i<8;i++) if ( abs( SH[i] ) < EPSILON ) SH[i]=0.0;
  /*  DISPOSAL */
  DI[1]=RM[16]*DIS[1]*P[9]/2000;
  DI[2]=RM[15]*DIS[1]*(1-P[9])/2000;
  DI[3]=RM[15]*DIS[2]/2000;
  DI[4]=DIS[1]*RM[14]/2000;
  for(i=1;i<5;i++) if ( abs( DI[i] ) < EPSILON ) DI[i]=0.0;
  /* PLASTICS PROCESSING */
```

```
PL[1]=PP[4]*L[2];
PL[2]=PL[1]+PP[3]+PP[1]+PP[2]*RM[5]/2000;
for(i=1;i<3;i++) if ( abs( PL[i] ) < EPSILON ) PL[i]=0.0;
/* RECYCLABLES MARKET */
if (PP[5] == 0.0)  PP[5]=1;  /*i.e. not processing the plastic*/
RE[1]=PP[5]*PR[1]*RM[7];
RE[2]=PP[5]*PR[2]*RM[9];
RE[3]=PP[5]*PR[3]*RM[10];
RE[4]=PR[4]*RM[11];
RE[5]=PR[6]*RM[13];
RE[6]=PR[5]*RM[12];
RE[7]=PR[7]*RM[2];
RE[8]=PR[8]*RM[3];
RE[9]=PP[5]*PR[9]*RM[5];
RE[10]=PR[10]*RM[4];
RE[11]=PR[11]*RM[6];
if (PR[12]==1) RE[9]=0.0;
if (PR[12]==0)
{
  RE[2]=0.0;
  RE[3]=0.0;
  RE[1]=0.0;
}
if (PR[13]==1) RE[10]=0.0;
if (PR[13]==0.0)
{
  RE[4]=0.0;
  RE[5]=0.0;
  RE[6]=0.0;
}
RE[12]=RE[1]+RE[2]+RE[3]+RE[4]+RE[5]+RE[6]+RE[7]
      +RE[8]+RE[9]+RE[10]+RE[11]+DI[2]+DI[3];
RE[13]=RM[16];
RE[14]=RE[4]+RE[5]+RE[6]+RE[10];
RE[15]=RE[1]+RE[2]+RE[3]+RE[9];
for(i=1;i<16;i++) if ( abs( RE[i] ) < EPSILON ) RE[i]=0.0;
/* COST AND REVENUE */
CR[1]=CS[6]*RM[15]/RM[1]*P[9]+(1-P[9])*CS[6];
CR[2]=(CS[6]-CR[1])*P[9];
CR[3]=(MH[2]+H[11])*RM[15]/RM[1]+MH[3]*P[9]+MH[4]*(1-P[9]);
CR[4]=(MH[4]-CR[3])*P[9];
CR[5]=RE[15]+RE[14]+RE[7]+RE[8]+RE[11]+DI[2]+DI[3];
CR[6]=CP[4]+CR[1]+CR[3]+SH[7]+PL[2];
CR[7]=CR[2]+SH[6]+CR[4]+DI[1];
CR[8]=RE[12]-CR[6]-CR[7];
CR[9]=RE[12]-DI[2]+DI[4];
for(i=1;i<10;i++) if ( abs( CR[i] ) < EPSILON ) CR[i]=0.0;
NC[1]=VOLO;
NC[2]=VOLA;
NC[3]=VOLG;
NC[4]=VOLP;
NC[5]=VOLN;
NC[6]=NC[1]+NC[2]+NC[3]+NC[4]+NC[5];
if(NC[6]==0.0)
```

```
{
  NC[8]=0.0;
  for(i=9;i<29;i++) NC[i]=0.0;
  screen_1();
}
NC[7]=VOLW;
if((NC[4]+NC[3]+NC[2])==0.0) NC[8]=NC[9]=NC[10]=0.0; /*  GOTO 22100 */
else
{
  NC[8]=CP[1]+CP[2]+P[7]*NC[4]/NC[6]+CP[3]*NC[4]/(NC[4]+NC[3]+NC[2]);
  NC[9]=P[7]*NC[3]/NC[6]+CP[3]*NC[3]/(NC[4]+NC[3]+NC[2]);
  NC[10]=P[7]*NC[2]/NC[6]+CP[3]*NC[2]/(NC[4]+NC[3]+NC[2]);
}
NC[11]= P[7]*NC[5]/NC[6];
NC[12]= P[7]*NC[1]/NC[6];
NC[13]=CR[1]*NC[4]/NC[6];
NC[14]=CR[1]*NC[3]/NC[6];
NC[15]=CR[1]*NC[2]/NC[6];
NC[16]=CR[1]*NC[5]/NC[6];
NC[17]=CR[1]*NC[1]/NC[6];
if((NC[4]+NC[3]+NC[2])==0.0) NC[18]=0.0;/*if == 0 then GOTO 22180*/
{
   NC[18]=H[2]*L[2]*H[3]*H[4]+H[5]*L[2]+H[6]*H[7]+H[1]*L[2]*NC[4]/
          (NC[4]+NC[3]+NC[2])+H[11]*NC[4]/NC[6];
}
if((NC[2]+NC[3])==0.0)
{
  NC[19]=NC[20]=0.0;
}
else
{
  if ((NC[4]+NC[3]+NC[2]) == 0.0)
  {
    NC[19]=NC[20]=0.0;
  }
  else
  {
    NC[19]=H[8]*L[2]+NC[3]/(NC[3]+NC[2])+H[1]*L[2]*NC[3]/
           (NC[4]+NC[3]+NC[2])+H[11]*NC[3]/NC[6];
    NC[20]=H[8]*L[2]+NC[2]/(NC[3]+NC[2])+H[1]*L[2]*NC[2]/
           (NC[4]+NC[3]+NC[2])+H[11]*NC[2]/NC[6];
  }
}
NC[21]=NC[5]*H[11]/NC[6];
NC[22]=NC[1]*H[11]/NC[6];
NC[23]=NC[8]+NC[13]+NC[18]+SH[4]+PL[2];
NC[24]=NC[9]+NC[14]+NC[19]+SH[2];
NC[25]=NC[10]+NC[15]+NC[20]+SH[1];
NC[26]=NC[11]+NC[16]+NC[21]+SH[3];
NC[27]=NC[12]+NC[17]+NC[22]+SH[5];
for(i=1;i<28;i++) if ( abs( NC[i] ) < EPSILON ) NC[i]=0.0;
if(RM[5]==0)
{
  NC[28]=0.0;
```

```
    screen_1();
  }
  else
  {
    NC[28]=NC[23]/RM[5];
    screen_1();
  }
  return(0);
}
```

```
/* FILE:READDATA.C */

#include "variable.h"

void readdata()
{/* READS DATA FROM ALL THE ACTIVE FILES and checks for 0 variables */
  int i,k;
  clrscr();
  frame_screen();
  gotoxy(1,1);
  printf("%c",ULDL);
  for(i=0;i<26;i++)printf("%c",HDL);
  printf("%c\n",URDL);
  printf("%c Reading default files:   %c\n",VDL,VDL);
  /* Find the last active regional data file */
  MODULE=1;
  INDEX=0;
  for(k=1;k<MAX_INDEX;k++)
  {
    if(STAR[MODULE][k] == 1) INDEX = k ;
    if(INDEX != 0)break;
  }
  if ( k >= MAX_INDEX )
  {
    gotoxy(10,5);
    printf("The Regional Data file is not completed");
    gotoxy(10,6);
    printf("Hit the space bar to return to the main menu.");
    k = get_key_1();
    if ( k == -32 ) main_menu();
    else
    {
      clrscr();
      printf("EXITING the program from readdata 1\n");
      sleep(DELAY);
      exit(0);
    }
  }
  printf("%c      %12s        %c\n",VDL,D[MODULE][INDEX],VDL);
  fil=fopen(D[MODULE][INDEX],"r");
  if ( fil == 0 ) flag_unopened_file( D[MODULE][INDEX] );
  for(i=1;i<5;i++) fscanf(fil,"%f",&R[i]);
  fclose(fil);
  /* Read policy data file.File #2 */
  /* Find the last active policy data file */
  MODULE=2;
  INDEX=0;
  for(k=1;k<MAX_INDEX;k++)
  {
    if(STAR[MODULE][k] == 1) INDEX = k ;
    if(INDEX != 0)break;
  }
  if ( k >= MAX_INDEX )
  {
```

```
    gotoxy(10,5);
    printf("The Policy Data file is not completed");
    gotoxy(10,6);
    printf("Hit the space bar to return to the main menu.");
    k = get_key_1();
    if ( k == -32 ) main_menu();
    else
    {
      clrscr();
      printf("EXITING the program from readdata 2\n");
      sleep(DELAY);
      exit(0);
    }
  }
  printf("%c      %12s        %c\n",VDL,D[MODULE][INDEX],VDL);
  fil=fopen(D[MODULE][INDEX],"r");
  if ( fil == 0 ) flag_unopened_file( D[MODULE][INDEX] );
  for(i=1;i<10;i++) fscanf(fil,"%f",&P[i]);
  fclose(fil);
  /* Read solid waste characteristics data file.File #3 */
  MODULE=3;
  INDEX=0;
  for(k=1;k<MAX_INDEX;k++)
  {
    if(STAR[MODULE][k] == 1) INDEX = k ;
    if(INDEX != 0)break;
  }
  if ( k >=  MAX_INDEX )
  {
    gotoxy(10,5);
    printf("The solid waste characteristics Data file is not completed");
    gotoxy(10,6);
    printf("Hit the space bar to return to the main menu.");
    k = get_key_1();
    if ( k == -32 ) main_menu();
    else
    {
      clrscr();
      printf("EXITING the program from readdata 3\n");
      sleep(DELAY);
      exit(0);
    }
  }
  printf("%c      %12s        %c\n",VDL,D[MODULE][INDEX],VDL);
  fil=fopen(D[MODULE][INDEX],"r");
  if ( fil == 0 ) flag_unopened_file( D[MODULE][INDEX] );
  for(i=1;i<20;i++) fscanf(fil,"%f",&W[i]);
  fclose(fil);
  /* Read Collection Equipment Characteristics data file.File #4 */
  MODULE=4;
  INDEX=0;
  for(k=1;k<MAX_INDEX;k++)
  {
    if(STAR[MODULE][k] == 1) INDEX = k ;
```

```
    if(INDEX != 0)break;
  }
  if ( k >= MAX_INDEX )
  {
    gotoxy(10,5);
    printf("The collection equipment characteristics Data file is not
completed");
    gotoxy(10,6);
    printf("Hit the space bar to return to the main menu.");
    k = get_key_1();
    if ( k == -32 ) main_menu();
    else
    {
      clrscr();
      printf("EXITING the program from readdata 4\n");
      sleep(DELAY);
      exit(0);
    }
  }
  printf("%c      %12s        %c\n",VDL,D[MODULE][INDEX],VDL);
  fil=fopen(D[MODULE][INDEX],"r");
  if ( fil == 0 ) flag_unopened_file( D[MODULE][INDEX] );
  for(i=1;i<33;i++)fscanf(fil,"%f",&C[i]);
  fclose(fil);
  /* Read Labor Costs data file.File #5 */
  MODULE=5;
  INDEX=0;
  for(k=1;k<MAX_INDEX;k++)
  {
    if(STAR[MODULE][k] == 1) INDEX = k ;
    if(INDEX != 0)break;
  }
  if ( k >= MAX_INDEX )
  {
    gotoxy(10,5);
    printf("The Labor Costs Data file is not completed");
    gotoxy(10,6);
    printf("Hit the space bar to return to the main menu.");
    k = get_key_1();
    if ( k == -32 ) main_menu();
    else
    {
      clrscr();
      printf("EXITING the program from readdata 5\n");
      sleep(DELAY);
      exit(0);
    }
  }
  printf("%c      %12s        %c\n",VDL,D[MODULE][INDEX],VDL);
  fil=fopen(D[MODULE][INDEX],"r");
  if ( fil == 0 ) flag_unopened_file( D[MODULE][INDEX] );
  for(i=1;i<3;i++) fscanf(fil,"%f",&L[i]);
  fclose(fil);
  /* Read Material Handling data file.File #6 */
```

```
MODULE=6;
INDEX=0;
for(k=1;k<MAX_INDEX;k++)
{
  if(STAR[MODULE][k] == 1) INDEX = k ;
  if(INDEX != 0)break;
}
if ( k >= MAX_INDEX )
{
  gotoxy(10,5);
  printf("The Material Handling Data file is not completed");
  gotoxy(10,6);
  printf("Hit the space bar to return to the main menu.");
  k = get_key_1();
  if ( k == -32 ) main_menu();
  else
  {
    clrscr();
    printf("EXITING the program from readdata 6\n");
    sleep(DELAY);
    exit(0);
  }
}
printf("%c      %12s        %c\n",VDL,D[MODULE][INDEX],VDL);
fil=fopen(D[MODULE][INDEX],"r");
if ( fil == 0 ) flag_unopened_file( D[MODULE][INDEX] );
for(i=1;i<12;i++) fscanf(fil,"%f",&H[i]);
fclose(fil);
/* Read Shipping Costs data file.File #7 */
MODULE=7;
INDEX=0;
for(k=1;k<MAX_INDEX;k++)
{
  if(STAR[MODULE][k] == 1) INDEX = k ;
  if(INDEX != 0)break;
}
if ( k >= MAX_INDEX )
{
  gotoxy(10,5);
  printf("The Shipping Costs Data file is not completed");
  gotoxy(10,6);
  printf("Hit the space bar to return to the main menu.");
  k = get_key_1();
  if ( k == -32 ) main_menu();
  else
  {
    clrscr();
    printf("EXITING the program from readdata 7\n");
    sleep(DELAY);
    exit(0);
  }
}
printf("%c      %12s        %c\n",VDL,D[MODULE][INDEX],VDL);
fil=fopen(D[MODULE][INDEX],"r");
```

```
if ( fil == 0 ) flag_unopened_file( D[MODULE][INDEX] );
for(i=1;i<19;i++) fscanf(fil,"%f",&S[i]);
fclose(fil);
/* Read Disposal Costs data file.File #8 */
MODULE=8;
INDEX=0;
for(k=1;k<MAX_INDEX;k++)
{
  if(STAR[MODULE][k] == 1) INDEX = k ;
  if(INDEX != 0)break;
}
if ( k >= MAX_INDEX )
{
  gotoxy(10,5);
  printf("The Disposal Costs Data file is not completed");
  gotoxy(10,6);
  printf("Hit the space bar to return to the main menu.");
  k = get_key_1();
  if ( k == -32 ) main_menu();
  else
  {
    clrscr();
    printf("EXITING the program from readdata 8\n");
    sleep(DELAY);
    exit(0);
  }
}
printf("%c      %12s        %c\n",VDL,D[MODULE][INDEX],VDL);
fil=fopen(D[MODULE][INDEX],"r");
if ( fil == 0 ) flag_unopened_file( D[MODULE][INDEX] );
for(i=1;i<3;i++) fscanf(fil,"%f",&DIS[i]);
fclose(fil);
/* Read Plastics Processing Costs data file.File #9 */
MODULE=9;
INDEX=0;
for(k=1;k<MAX_INDEX;k++)
{
  if(STAR[MODULE][k] == 1) INDEX = k ;
  if(INDEX != 0)break;
}
if ( k >= MAX_INDEX )
{
  gotoxy(10,5);
  printf("The Plastics Processing Costs Data file is not completed");
  gotoxy(10,6);
  printf("Hit the space bar to return to the main menu.");
  k = get_key_1();
  if ( k == -32 ) main_menu();
  else
  {
    clrscr();
    printf("EXITING the program from readdata 9\n");
    sleep(DELAY);
    exit(0);
```

```
    }
  }
  printf("%c      %12s        %c\n",VDL,D[MODULE][INDEX],VDL);
  fil=fopen(D[MODULE][INDEX],"r");
  if ( fil == 0 ) flag_unopened_file( D[MODULE][INDEX] );
  for(i=1;i<6;i++) fscanf(fil,"%f",&PP[i]);
  fclose(fil);
  /* Read Selling Prices of Recycled Waste Material Data file.File #10 */
  MODULE=10;
  INDEX=0;
  for(k=1;k<MAX_INDEX;k++)
  {
    if(STAR[MODULE][k] == 1) INDEX = k ;
    if(INDEX != 0)break;
  }
  if ( k >= MAX_INDEX )
  {
    gotoxy(10,5);
    printf("The Selling Prices of Recycled Waste Material Data file is
not completed");
    gotoxy(10,6);
    printf("Hit the space bar to return to the main menu.");
    k = get_key_1();
    if ( k == -32 ) main_menu();
    else
    {
      clrscr();
      printf("EXITING the program from readdata 10\n");
      sleep(DELAY);
      exit(0);
    }
  }
  printf("%c      %12s        %c\n",VDL,D[MODULE][INDEX],VDL);
  fil=fopen(D[MODULE][INDEX],"r");
  if ( fil == 0 ) flag_unopened_file( D[MODULE][INDEX] );
  for(i=1;i<14;i++) fscanf(fil,"%f",&PR[i]);
  fclose(fil);
  /* Read  Collection Point data file.File #11 */
  MODULE=11;
  INDEX=0;
  for(k=1;k<MAX_INDEX;k++)
  {
    if(STAR[MODULE][k] == 1) INDEX = k ;
    if(INDEX != 0)break;
  }
  if ( k >= MAX_INDEX )
  {
    gotoxy(10,5);
    printf("The Collection Point Data file is not completed");
    gotoxy(10,6);
    printf("Hit the space bar to return to the main menu.");
    k = get_key_1();
    if ( k == -32 ) main_menu();
    else
```

```
    {
      clrscr();
      printf("EXITING the program from readdata 11\n");
      sleep(DELAY);
      exit(0);
    }
  }
  printf("%c      %12s        %c\n",VDL,D[MODULE][INDEX],VDL);
  fil=fopen(D[MODULE][INDEX],"r");
  if ( fil == 0 ) flag_unopened_file( D[MODULE][INDEX] );
  for(i=1;i<3;i++) fscanf(fil,"%f",&PT[i]);
  fclose(fil);
  printf("%c",BLDL);
  for(i=0;i<26;i++)printf("%c",HDL);
  printf("%c",BRDL);
  sleep(DELAY);
  return;
}

void modules()
{ /*
    input data file modification (11 data files )
    also known as the secondary menu in the basic version
  */
  int f;
  clrscr();
  frame_screen();
  gotoxy(19,3); printf("Which module would you like to modify?");
  gotoxy(20,6); printf(" (1)  Regional Data");
  gotoxy(20,7); printf(" (2)  Policy Data");
  gotoxy(20,8); printf(" (3)  Solid Waste Characteristics");
  gotoxy(20,9); printf(" (4)  Collection Equipment Characteristics");
  gotoxy(20,11);printf(" (5)  Labor Costs");
  gotoxy(20,12);printf(" (6)  Material Handling ");
  gotoxy(20,13);printf(" (7)  Shipping Costs");
  gotoxy(20,14);printf(" (8)  Disposal Costs");
  gotoxy(20,16);printf(" (9)  Plastics Processing Costs");
  gotoxy(20,17);printf("  (10)  Selling  Prices  of  Recycled  Waste
Material");
  gotoxy(20,18);printf(" (11) Collection Point Data");
  gotoxy(20,19);printf(" ESC  Back to main menu");
  gotoxy(20,21);printf(" NOTE: Enter from the keyboard: 1, 2, 3, ... , or
11");
  gotoxy(20,22);printf("       along with a <CR>.");
  gotoxy(58,3);
  f=get_key_2();
  if ( f == 0 ) MODULE = v;
  else if ( f == -27 ) main_menu();
  switch( MODULE )
  {
    case 1 :
             gotoxy(59,3);
             printf("              ");
             gotoxy(59,3);
```

```
            printf("%d",MODULE);
            gotoxy(59,3);
            regional();
            break;
 case 2 :
            gotoxy(59,3);
            printf("              ");
            gotoxy(59,3);
            printf("%d",MODULE);
            gotoxy(59,3);
            policy();
            break;
 case 3 :
            gotoxy(59,3);
            printf("              ");
            gotoxy(59,3);
            printf("%d",MODULE);
            gotoxy(59,3);
            solidwas();
            break;
 case 4 :
            gotoxy(59,3);
            printf("              ");
            gotoxy(59,3);
            printf("%d",MODULE);
            gotoxy(59,3);
            collect();
            break;
 case 5 :
            gotoxy(59,3);
            printf("              ");
            gotoxy(59,3);
            printf("%d",MODULE);
            gotoxy(59,3);
            labor();
            break;
 case 6:
            gotoxy(59,3);
            printf("              ");
            gotoxy(59,3);
            printf("%d",MODULE);
            gotoxy(59,3);
            material();
            break;
 case 7:
            gotoxy(59,3);
            printf("              ");
            gotoxy(59,3);
            printf("%d",MODULE);
            gotoxy(59,3);
            shipping();
            break;
 case 8:
            gotoxy(59,3);
```

```
                printf("                ");
                gotoxy(59,3);
                printf("%d",MODULE);
                gotoxy(59,3);
                disposal();
                break;
        case 9:
                gotoxy(59,3);
                printf("                ");
                gotoxy(59,3);
                printf("%d",MODULE);
                gotoxy(59,3);
                plasproc();
                break;
        case 10:
                gotoxy(59,3);
                printf("                ");
                gotoxy(59,3);
                printf("%d",MODULE);
                gotoxy(59,3);
                selling();
                break;
        case 11:
                gotoxy(59,3);
                printf("                ");
                gotoxy(59,3);
                printf("%d",MODULE);
                gotoxy(59,3);
                colpoint();
                break;
        default:
                gotoxy(60,4);
                printf("Try again ...");
                sleep(DELAY);
                modules();
                gotoxy(59,3);
                printf("                ");
                gotoxy(60,4);
                printf("                ");
                gotoxy(59,3);
                break;
    }
}
```

```
/* FILE:OUTPUT.c */

#include "variable.h"

void screen_1()
{
  int c;
  clrscr();
  frame_screen();
  gotoxy(32,1);printf(" Output Screen 1 ");
  gotoxy(5,3);printf("Revenue, Subsidies, and Credits ... $/month");
  gotoxy(10,5);printf("Plastics.....%9.2f",RE[15]);
  gotoxy(10,6);printf("Glass.....%9.2f",RE[14]);
  gotoxy(10,7);printf("Aluminum.....%9.2f",RE[7]);
  gotoxy(10,8);printf("Newspaper.....%9.2f",RE[8]);
  gotoxy(10,9);printf("Other Material.....%9.2f",RE[11]);
  gotoxy(10,10);printf("Tipping Fee Credit (wt based).....%9.2f",DI[2]);
  gotoxy(10,11);printf("Government Subsidies.....%9.2f",DI[3]);
  gotoxy(10,12);
  printf("-------------------------------------------------------------");
  gotoxy(10,13);printf("Total Revenue ..... ");
  gotoxy(57,13);printf("%9.2f",RE[12]);
  gotoxy(10,16);printf("Tipping Fee Credit (volume based)..%9.2f",DI[4]);
  gotoxy(10,18);printf("Adjusted Total ..... ");
  gotoxy(57,18);printf("%9.2f",CR[9]);
  gotoxy(10,19);printf("(not used in subsequent calculations)");
  gotoxy(5,21);printf("(Hit space bar to continue.)");
  gotoxy(5,22);printf("(Hit ESCape to return to the Main Menu.)");
wait_until_esc:
  c = get_key_1();
  if      ( c == -27 ) main_menu();
  else if ( c == -32 ) screen_2();
  else  goto wait_until_esc ;
  return;
}

void screen_2()
{
  int c;
  clrscr();
  frame_screen();
  gotoxy(32,1);printf(" Output Screen 2 ");
  gotoxy(5,3);printf("Recycling Costs ... $/month");
  gotoxy(15,5);printf("Collection Point.....%9.2f",CP[4]);
  gotoxy(15,6);printf("Collection.....%9.2f",CR[1]);
  gotoxy(15,7);printf("Material Handling.....%9.2f",CR[3]);
  gotoxy(15,8);printf("Shipping.....%9.2f",SH[7]);
  gotoxy(15,9);printf("Plastics Processing.....%9.2f",PL[2]);
  gotoxy(15,10);
  printf("-----------------------------------------------------");
  gotoxy(15,11);printf("Total Recycling Costs ..... ");
  gotoxy(58,11);printf("%9.2f",CR[6]);
  gotoxy(5,15);printf("(Hit space bar to continue.)");
  gotoxy(5,16);printf("(Hit U or u to view previous screen.)");
```

```
  gotoxy(5,17);printf("(Hit ESCape to return to the Main Menu.)");
wait_until_esc:
  c = get_key_1();
  if      ( c == -27 ) main_menu();
  else if ( c == -32 ) screen_3();
  else if ( c == -85 ) screen_1();
  else  goto wait_until_esc ;
  return;
}

void screen_3()
{
  int c;
  clrscr();
  frame_screen();
  gotoxy(32,1);printf(" Output Screen 3 ");
  gotoxy(5,3);printf("Nonrecyclable Costs ... $/month");
  gotoxy(15,5);printf("Collection.....%9.2f",CR[2]);
  gotoxy(15,6);printf("Material Handling.....%9.2f",CR[4]);
  gotoxy(15,7);printf("Shipping.....%9.2f",SH[6]);
  gotoxy(15,8);printf("Tipping Fees.....%9.2f",DI[1]);
  gotoxy(15,9);
  printf("-----------------------------------------------------");
  gotoxy(15,10);printf("Total Nonrecyclable Cost ..... ");
  gotoxy(58,10);printf("%9.2f",CR[7]);
  gotoxy(5,13);printf("Net Revenue ... $/month");
  gotoxy(58,13);printf("%9.2f",CR[8]);
  BETF=-(CR[8]-DI[2])/RM[15]*2000;
  gotoxy(5,16);printf("Break Even Tipping Fee ... $/ton (Net Revenue =
$0.00)");
  gotoxy(58,16);printf("%9.2f",BETF);
  gotoxy(5,20);printf("(Hit space bar to continue.)");
  gotoxy(5,21);printf("(Hit U or u to view previous screen.)");
  gotoxy(5,22);printf("(Hit ESCape to return to the Main Menu.)");
wait_until_esc:
  c = get_key_1();
  if      ( c == -27 ) main_menu();
  else if ( c == -32 ) screen_4();
  else if ( c == -85 ) screen_2();
  else  goto wait_until_esc ;
  return;
}

void screen_4()
{
  int c;
  clrscr();
  frame_screen();
  gotoxy(32,1);printf(" Output Screen 4 ");
  gotoxy(27,3);printf("Cost Breakdown ... $/month");
  gotoxy(22,5);printf("Plastic");
  gotoxy(35,5);printf("Glass");
  gotoxy(45,5);printf("Aluminum");
  gotoxy(57,5);printf("Newspaper");
```

```
   gotoxy(72,5);printf("Other");
   gotoxy(2,6);printf("Collection Point");
   gotoxy(20,6);printf("%9.2f",NC[8]);
   gotoxy(32,6);printf("%9.2f",NC[9]);
   gotoxy(44,6);printf("%9.2f",NC[10]);
   gotoxy(56,6);printf("%9.2f",NC[11]);
   gotoxy(68,6);printf("%9.2f",NC[12]);
   gotoxy(2,8);printf("Collection System");
   gotoxy(20,8);printf("%9.2f",NC[13]);
   gotoxy(32,8);printf("%9.2f",NC[14]);
   gotoxy(44,8);printf("%9.2f",NC[15]);
   gotoxy(56,8);printf("%9.2f",NC[16]);
   gotoxy(68,8);printf("%9.2f",NC[17]);
   gotoxy(2,10);printf("Material Handling");
   gotoxy(20,10);printf("%9.2f",NC[18]);
   gotoxy(32,10);printf("%9.2f",NC[19]);
   gotoxy(44,10);printf("%9.2f",NC[20]);
   gotoxy(56,10);printf("%9.2f",NC[21]);
   gotoxy(68,10);printf("%9.2f",NC[22]);
   gotoxy(2,12);printf("Shipping");
   gotoxy(20,12);printf("%9.2f",SH[4]);
   gotoxy(32,12);printf("%9.2f",SH[2]);
   gotoxy(44,12);printf("%9.2f",SH[1]);
   gotoxy(56,12);printf("%9.2f",SH[3]);
   gotoxy(68,12);printf("%9.2f",SH[5]);
   gotoxy(2,14);printf("Processing");
   gotoxy(20,14);printf("%9.2f",PL[2]);
   gotoxy(33,14);printf("    0.00        0.00        0.00        0.00");
   gotoxy(2,15);
   printf("------------------------------------------------------------");
   gotoxy(2,16);printf("Total");
   gotoxy(20,16);printf("%9.2f",NC[23]);
   gotoxy(32,16);printf("%9.2f",NC[24]);
   gotoxy(44,16);printf("%9.2f",NC[25]);
   gotoxy(56,16);printf("%9.2f",NC[26]);
   gotoxy(68,16);printf("%9.2f",NC[27]);
   gotoxy(20,19);printf("Plastic Collection Cost ($/lb) %9.2f",NC[28]);
   gotoxy(5,21);printf("(Hit space bar to continue.)");
   gotoxy(5,22);printf("(Hit U or u to view previous screen.)");
   gotoxy(5,23);printf("(Hit ESCape to return to the Main Menu.)");
wait_until_esc:
   c = get_key_1();
   if      ( c == -27 ) main_menu();
   else if ( c == -32 ) screen_5();
   else if ( c == -85 ) screen_3();
   else  goto wait_until_esc ;
   return;
}

void screen_5()
{
   int c;
   clrscr();
   frame_screen();
```

```
   gotoxy(32,1);printf(" Output Screen 5 ");
   gotoxy(51,3);printf("Wt.(lbs)");
   gotoxy(62,3);printf("VOL.(YARDS^3)");
   gotoxy(5,4);printf("Total Waste Stream per month");
   gotoxy(48,4);printf("%9.2f",RM[1]);
   gotoxy(62,4);printf("%9.2f",VOLW);
   gotoxy(15,6);printf("Recycled Plastic .................");
   printf("%9.2f",RM[5]);
   gotoxy(62,6);printf("%9.2f",VOLP);
   gotoxy(15,7);printf("Recycled Glass ...................");
   printf("%9.2f",RM[4]);gotoxy(62,7);printf("%9.2f",VOLG);
   gotoxy(15,8);printf("Recycled Newspaper ..............%9.2f",RM[3]);
   gotoxy(62,8);printf("%9.2f",VOLN);
   gotoxy(15,9);printf("Recycled Aluminum ...............%9.2f",RM[2]);
   gotoxy(62,9);printf("%9.2f",VOLA);
   gotoxy(15,10);printf("Other Recycled Material ..........%9.2f",RM[6]);
   gotoxy(62,10);printf("%9.2f",VOLO);
   gotoxy(15,11);
   printf("-------------------------------------------------------------");
   gotoxy(5,12);printf("Total recyclables removed from waste stream   ");
   gotoxy(49,12);printf("%9.2f",RM[15]);
   gotoxy(62,12);printf("%9.2f",VOLTOT);
   gotoxy(4,14);printf("%% of total waste stream");
   gotoxy(49,14);printf("%9.2f%%",RM[15]*100/RM[1]);
   gotoxy(62,14);printf("%9.2f%%",VOLTOT*100/VOLW);
   gotoxy(5,16);printf("(Hit D or d to save output on disk.)");
   gotoxy(5,17);printf("(Hit P or p for a print-out of the output.)");
   gotoxy(5,18);printf("(Hit U or u to view previous screen.)");
   gotoxy(5,20);printf("(Hit ESCape to return to the Main Menu.)");
wait_until_esc:
   c = get_key_1();
   if      ( c == -27 ) main_menu();
   else if ( c == -68 )
   {
     display_jimbo();
     get_file_name();
     save_rpt();
     update_jimbo();
   }
   else if ( c == -80 )
   {
     display_jimbo();
     get_file_name();
     save_rpt();
     scr_print();
     update_jimbo();
   }
   else if ( c == -85 ) screen_4();
   else  goto wait_until_esc ;
   return;
}
```

```
/* FILE:HARDCOPY.C */

#include "variable.h"

void check_closing ( t )
int t;
{
  if ( t == 0 ) return;
  else
  {
    printf("\nError closing file");
    sleep(DELAY);
  };
  return;
}

void prt_regional()
{
  int i;
  char ARA[13];
  FILE *ara;
  fil = fopen(D[MODULE][INDEX],"r");
  if ( fil == 0 ) flag_unopened_file(D[MODULE][INDEX]);
  else
  {
    for(i=1;i<5;i++)
    {
      R[i]=0.0;
      fgets(B,MFL,fil);
      sscanf(B,"%f",&R[i]);
    };
    sprintf(ARA,"TEMP.PRT");
    ara = fopen(ARA,"w");
    if ( ara == 0 ) flag_unopened_file(ARA);
    else
    {
      fprintf(ara,"\n       Regional   Data   from   %s   file\n\n\n",
D[MODULE][INDEX] );
      fprintf(ara,"The  size  of  the  region  of  interest  (square
miles)...%9.2f\n",R[1]);
      fprintf(ara,"Population            density            (people/sqr
mile).............%9.2f\n",R[2]);
      fprintf(ara,"Route  miles  (total  miles  in  one  collection
cycle)...%9.2f\n",R[3]);
      fprintf(ara,"Number                    of                  collection
points.....................%9.2f\n",R[4]);
      fprintf(ara,"%c",12);
      check_closing ( fclose(ara) );
      clrscr();gotoxy(1,1);
      sprintf(ARA,"COPY TEMP.PRT PRN:");
      check_status ( system( ARA ));
      sleep(DELAY);
      unlink("TEMP.PRT");
    };
```

```
   };
   fclose(fil);
   return;
 }

 void prt_policy()
 { int i;
   char ARA[13];
   FILE *ara;
   fil = fopen(D[MODULE][INDEX],"r");
   if ( fil == 0 ) flag_unopened_file(D[MODULE][INDEX]);
   else
   {
     for(i=1;i<10;i++)
     {
       P[i]=0.0;
       fgets(B,MFL,fil);
       sscanf(B,"%f",&P[i]);
     };
     sprintf(ARA,"TEMP.PRT");
     ara = fopen(ARA,"w");
     if ( ara == 0 ) flag_unopened_file(ARA);
     else
     {
       fprintf(ara,"\n    Policy Data from file %s\n\n",D[MODULE][INDEX]);
       fprintf(ara,"Compliance rate (0-100) for:\n");
       fprintf(ara,"Aluminum.....%9.2f\n",P[1]);
       fprintf(ara,"Newspaper.....%9.2f\n",P[2]);
       fprintf(ara,"Glass.....%9.2f\n",P[3]);
       fprintf(ara,"Plastics.....%9.2f\n",P[4]);
       f p r i n t f ( a r a , " O t h e r
Material.....%9.2f\n",P[5]);
       fprintf(ara,"Pay-out rate for plastics
($/lb).....%9.2f\n",P[6]);
       fprintf(ara,"Monthly public awareness program
cost.....%9.2f\n",P[7]);
       fprintf(ara,"Number of collection cycles per
month.....%9.2f\n",P[8]);
       fprintf(ara,"Are recyclables and solid waste \n");
       fprintf(ara,"collected simultaneously? (1-yes/0-
no)....%9.2f\n",P[9]);
       fprintf(ara,"%c",12);
       check_closing(fclose(ara));
       clrscr(); gotoxy(1,1);
       sprintf(ARA,"COPY TEMP.PRT PRN:");
       check_status(system(ARA));
       sleep(DELAY);
       unlink("TEMP.PRT");
     };
   };
   fclose(fil);
   return;
 }
```

```
void prt_solidwas()
{ int i;
  char ARA[13];
  FILE *ara;
  fil = fopen(D[MODULE][INDEX],"r");
  if ( fil == 0 ) flag_unopened_file(D[MODULE][INDEX]);
  else
  {
    for(i=1;i<20;i++)
    {
      W[i]=0.0;
      fgets(B,MFL,fil);
      sscanf(B,"%f",&W[i]);
    }
    sprintf(ARA,"TEMP.PRT");
    ara = fopen(ARA,"w");
    if ( ara == 0 ) flag_unopened_file(ARA);
    else
    {
      fprintf(ara,"\n                Solid   waste   Data   from   file
%s\n\n",D[MODULE][INDEX]);
      fprintf(ara," Solid Waste Characteristics (screen 1) \n\n");
      fprintf(ara,"(Most   values   in   this   module   are   universal   and
will\n");
      fprintf(ara,"not   vary   across   different   collection   system
types.)\n");
      fprintf(ara,"Average amount of solid waste created\n");
      fprintf(ara,"per   person   per   month   (lbs/month).....
%12.2f\n",W[1]);
      fprintf(ara,"Percentage (0-100) of soid waste that is:\n");
      fprintf(ara,"Aluminum.....
%12.2f\n",W[2]);
      fprintf(ara,"Newspaper.....
%12.2f\n",W[3]);
      fprintf(ara,"Glass.....
%12.2f\n",W[4]);
      fprintf(ara,"Plastics.....
%12.2f\n",W[5]);
      fprintf(ara,"Other   material   of   interest.....
%12.2f\n",W[6]);
      fprintf(ara,"Percentage (0-100) of plastics that are:\n");
      fprintf(ara,"PE.....
%12.2f\n",W[7]);
      fprintf(ara,"PET.....
%12.2f\n\n",W[8]);
      fprintf(ara," Solid Waste Characteristics (screen 2)\n\n");
      fprintf(ara,"Percentage (0-100) of PET that is:\n");
      fprintf(ara,"Clear.....
%12.2f\n",W[9]);
      fprintf(ara,"Colored.....
%12.2f\n",W[10]);
      fprintf(ara,"Percentage (0-100) of glass that is:\n");
      fprintf(ara,"Clear.....
%12.2f\n",W[11]);
```

```
      fprintf(ara,"Green.....
%12.2f\n",W[12]);
      fprintf(ara,"Brown.....
%12.2f\n",W[13]);
      fprintf(ara,"Average density (lb/ft^3) of solid waste....
%12.2f\n",W[14]);
      fprintf(ara,"Average density (lb/ft^3) of recyclable:\n");
      fprintf(ara,"Aluminum.....
%12.2f\n",W[15]);
      fprintf(ara,"Newspaper.....
%12.2f\n",W[16]);
      fprintf(ara,"Glass.....
%12.2f\n",W[17]);
      fprintf(ara,"Plastic.....
%12.2f\n",W[18]);
      fprintf(ara,"Other          Material.....
%12.2f\n",W[19]);
      fprintf(ara,"%c",12);
      check_closing(fclose(ara));
      clrscr(); gotoxy(1,1);
      sprintf(ARA,"COPY TEMP.PRT PRN:");
      check_status(system(ARA));
      sleep(DELAY);
      unlink("TEMP.PRT");
    }
  }
  fclose(fil);
  return;
}

void prt_collect()
{
  int i;
  char ARA[13];
  FILE *ara;
  fil = fopen(D[MODULE][INDEX],"r");
  if ( fil == 0 ) flag_unopened_file(D[MODULE][INDEX]);
  else
  {
    for(i=1;i<33;i++)
    {
      C[i]=0.0;
      fgets(B,MFL,fil);
      sscanf(B,"%f",&C[i]);
    }
    sprintf(ARA,"TEMP.PRT");
    ara = fopen(ARA,"w");
    if ( ara == 0 ) flag_unopened_file(ARA);
    else
    {
      fprintf(ara,"\n Collection Equipment Data from %s file\n\n\n",
D[MODULE][INDEX] );
      fprintf(ara,"\nCollection Equipment");
```

```
      fprintf(ara,"\nVehicle Type                    1          2
3          4");
      fprintf(ara,"\n-------------------------------------------------------");
      fprintf(ara,"\n# of Units           ");
      for(i=1;i<26;i+=8) fprintf(ara,"%12.2f",C[i]);
      fprintf(ara,"\nWeight Capacity (lbs) ");
      for(i=2;i<27;i+=8) fprintf(ara,"%12.2f",C[i]);
      fprintf(ara,"\nVolume Capacity (ft^3)");
      for(i=3;i<28;i+=8) fprintf(ara,"%12.2f",C[i]);
      fprintf(ara,"\nEquipment Cost");
      fprintf(ara,"\n/month/unit          ");
      for(i=4;i<29;i+=8) fprintf(ara,"%12.2f",C[i]);
      fprintf(ara,"\nOverhead");
      fprintf(ara,"\n/month/unit          ");
      for(i=5;i<30;i+=8) fprintf(ara,"%12.2f",C[i]);
      fprintf(ara,"\nCost/mile/unit      ");
      for(i=6;i<31;i+=8) fprintf(ara,"%12.2f",C[i]);
      fprintf(ara,"\nStaff/unit (frac ok)  ");
      for(i=7;i<32;i+=8) fprintf(ara,"%12.2f",C[i]);
      fprintf(ara,"\nAvg # Collect");
      fprintf(ara,"\npoints/month/unit    ");
      for(i=8;i<33;i+=8) fprintf(ara,"%12.2f",C[i]);
      fprintf(ara,"%c",12);
      check_closing(fclose(ara));
      clrscr();
      gotoxy(1,1);
      sprintf(ARA,"COPY TEMP.PRT PRN:");
      check_status(system(ARA));
      sleep(DELAY);
      unlink("TEMP.PRT");
    }
  }
  fclose(fil);
  return;
}

void prt_labor()
{ int i;
  char ARA[13];
  FILE *ara;
  fil = fopen(D[MODULE][INDEX],"r");
  if ( fil == 0 ) flag_unopened_file(D[MODULE][INDEX]);
  else
  {
    sprintf(ARA,"TEMP.PRT");
    ara = fopen(ARA,"w");
    if ( ara == 0 ) flag_unopened_file(ARA);
    else
    {
      for(i=1;i<3;++i)
      {
        L[i]=0.0;
        fgets(B,MFL,fil);
        sscanf(B,"%f",&L[i]);
```

```
      }
      fprintf(ara,"\n   Labor   Costs   Data   from   %s   file\n\n\n",
D[MODULE][INDEX] );
      fprintf(ara,"\nLabor Costs");
      fprintf(ara,"\nThe average monthly cost per laborer for:\n");
      fprintf(ara,"\nCollectors.............%12.2f",L[1]);
      fprintf(ara,"\nMaterial Handlers......%12.2f",L[2]);
      fprintf(ara,"%c",12);
      check_closing(fclose(ara));
      clrscr(); gotoxy(1,1);
      sprintf(ARA,"COPY TEMP.PRT PRN:");
      check_status(system(ARA));
      sleep(DELAY);
      unlink("TEMP.PRT");
    }
  }
  fclose(fil);
  return;
}

void prt_material()
{ int i;
  char ARA[13];
  FILE *ara;
  fil = fopen(D[MODULE][INDEX],"r");
  if ( fil == 0 ) flag_unopened_file(D[MODULE][INDEX]);
  else
  {
    sprintf(ARA,"TEMP.PRT");
    ara = fopen(ARA,"w");
    if ( ara == 0 ) flag_unopened_file(ARA);
    else
    {
      for(i=1;i<12;i++)
      {
        H[i]=0.0;
        fgets(B,MFL,fil);
        sscanf(B,"%f",&H[i]);
      }
      fprintf(ara,"\n Material Handling Costs Data from %s file\n\n\n",
D[MODULE][INDEX] );
      fprintf(ara,"\n Material Handling\n");
      fprintf(ara,"\n                              # operators   # machines
equip cost");
      f   p   r   i   n   t   f   (   a   r   a   ,   "   \   n
($/month/unit)");
      fprintf(ara,"\n-------------------------------------------------------");
      fprintf(ara,"\nSorting-manual                  %12.2f            n/a
n/a",H[1]);
      fprintf(ara,"\nGranulation/Shredding     ");
      for(i=1;i<4;i++) fprintf(ara," %12.2f",H[i+1]);
      fprintf(ara,"\nBaling                    ");
      for(i=1;i<4;i++) fprintf(ara," %12.2f",H[i+4]);
      fprintf(ara,"\nCrushing (glass/cans only)");
```

```
      for(i=1;i<4;i++) fprintf(ara," %12.2f",H[i+7]);
      fprintf(ara,"\n\nMonthly overhead for the material handling
facility....%12.2f",H[11]);
      fprintf(ara,"%c",12);
      check_closing(fclose(ara));
      clrscr(); gotoxy(1,1);
      sprintf(ARA,"COPY TEMP.PRT PRN:");
      check_status(system(ARA));
      sleep(DELAY);
      unlink("TEMP.PRT");
    }
  }
  fclose(fil);
  return;
}

void prt_shipping()
{ int i;
  char ARA[13];
  FILE *ara;
  fil = fopen(D[MODULE][INDEX],"r");
  if ( fil == 0 ) flag_unopened_file(D[MODULE][INDEX]);
  else
  {
    sprintf(ARA,"TEMP.PRT");
    ara = fopen(ARA,"w");
    if ( ara == 0 ) flag_unopened_file(ARA);
    else
    {
      for(i=1;i<19;i++)
      {
        S[i]=0.0;
        fgets(B,MFL,fil);
        sscanf(B,"%f",&S[i]);
      }
      fprintf(ara,"\n Shipping Costs (screen 1) Data from %s file\n\n\n",
D[MODULE][INDEX] );
      fprintf(ara,"\nDistance (miles) from the materials handling
facility to:");
      fprintf(ara,"\nGlass market..... %12.2f",S[1]);
      fprintf(ara,"\nNewspaper market..... %12.2f",S[2]);
      fprintf(ara,"\nAluminum market..... %12.2f",S[3]);
      fprintf(ara,"\nPlastic processing plant..... %12.2f",S[4]);
      fprintf(ara,"\nLandfill..... %12.2f",S[5]);
      fprintf(ara,"\nMarket for other materials... %12.2f",S[6]);
      fprintf(ara,"\nCost ($) per mile per shipment for:");
      fprintf(ara,"\nGlass..... %12.2f",S[7]);
      fprintf(ara,"\nNewspaper..... %12.2f",S[8]);
      fprintf(ara,"\nAluminum..... %12.2f",S[9]);
      fprintf(ara,"\nPlastic..... %12.2f",S[10]);
      fprintf(ara,"\nSolid waste..... %12.2f",S[11]);
      fprintf(ara,"\nOther material..... %12.2f",S[12]);
      fprintf(ara,"\n\n Shipping Costs (screen 2)\n");
      fprintf(ara,"\nWeight capacity (lbs) per shipment of:");
```

```
      fprintf(ara,"\nGlass..... %12.2f",S[13]);
      fprintf(ara,"\nNewspaper..... %12.2f",S[14]);
      fprintf(ara,"\nAluminum..... %12.2f",S[15]);
      fprintf(ara,"\nPlastic..... %12.2f",S[16]);
      fprintf(ara,"\nSolid waste..... %12.2f",S[17]);
      fprintf(ara,"\nOther material..... %12.2f",S[18]);
      fprintf(ara,"%c",12);
      check_closing(fclose(ara));
      clrscr(); gotoxy(1,1);
      sprintf(ARA,"COPY TEMP.PRT PRN:");
      check_status(system(ARA));
      sleep(DELAY);
      unlink("TEMP.PRT");
    }
  }
  fclose(fil);
  return;
}

void prt_disposal()
{ int i;
  char ARA[13];
  FILE *ara;
  fil = fopen(D[MODULE][INDEX],"r");
  if ( fil == 0 ) flag_unopened_file(D[MODULE][INDEX]);
  else
  {
    sprintf(ARA,"TEMP.PRT");
    ara = fopen(ARA,"w");
    if ( ara == 0 ) flag_unopened_file(ARA);
    else
    {
      for(i=1;i<3;i++)
      {
        DIS[i]=0.0;
        fgets(B,MFL,fil);
        sscanf(B,"%f",&DIS[i]);
      }
      fprintf(ara,"\n  Disposal  Costs  Data  from  %s  file\n\n\n",
D[MODULE][INDEX] );
      fprintf(ara,"\nDisposal Costs");
      fprintf(ara,"\nTipping      fee      ($/ton).....
%12.2f",DIS[1]);
      fprintf(ara,"\nGovernment   subsidy   or   abatement   ($/ton)..
%12.2f",DIS[2]);
      fprintf(ara,"%c",12);
      check_closing(fclose(ara));
      clrscr(); gotoxy(1,1);
      sprintf(ARA,"COPY TEMP.PRT PRN:");
      check_status(system(ARA));
      sleep(DELAY);
      unlink("TEMP.PRT");
    }
  }
```

```
  fclose(fil);
  return;
}

void prt_plasproc()
{ int i;
  char ARA[13];
  FILE *ara;
  fil = fopen(D[MODULE][INDEX],"r");
  if ( fil == 0 ) flag_unopened_file(D[MODULE][INDEX]);
  else
  {
    sprintf(ARA,"TEMP.PRT");
    ara = fopen(ARA,"w");
    if ( ara == 0 ) flag_unopened_file(ARA);
    else
    {
      for(i=1;i<6;i++)
      {
        PP[i]=0.0;
        fgets(B,MFL,fil);
        sscanf(B,"%f",&PP[i]);
      }
      fprintf(ara,"\n Plastics Processing Costs Data from %s file\n\n\n",
D[MODULE][INDEX] );
      fprintf(ara,"\nEquipment                cost/month.....................
%12.2f",PP[1]);
      fprintf(ara,"\nDirect     process     costs     ($/ton).....
%12.2f",PP[2]);
      fprintf(ara,"\nProcessing     facility     monthly     overhead..
%12.2f",PP[3]);
      fprintf(ara,"\n=    of    operators    at    processing    plant....
%12.2f",PP[4]);
      fprintf(ara,"\nProcess     efficiency     ratio     (0-1).....
%12.2f",PP[5]);
      fprintf(ara,"%c",12);
      check_closing(fclose(ara));
      clrscr(); gotoxy(1,1);
      sprintf(ARA,"COPY TEMP.PRT PRN:");
      check_status(system(ARA));
      sleep(DELAY);
      unlink("TEMP.PRT");
    }
  }
  fclose(fil);
  return;
}

void prt_selling()
{ int i;
  char ARA[13];
  FILE *ara;
  fil = fopen(D[MODULE][INDEX],"r");
  if ( fil == 0 ) flag_unopened_file(D[MODULE][INDEX]);
```

```
   else
   {
     sprintf(ARA,"TEMP.PRT");
     ara = fopen(ARA,"w");
     if ( ara == 0 ) flag_unopened_file(ARA);
     else
     {
       for(i=1;i<14;i++)
       {
         PR[i]=0.0;
         fgets(B,MFL,fil);
         sscanf(B,"%f",&PR[i]);
       }
       fprintf(ara,"\n  Selling  Prices  Data  from  %s  file\n\n\n",
D[MODULE][INDEX] );
       fprintf(ara,"\n Selling Prices of Recycled Waste Material ");
       fprintf(ara,"\nPrice per lb of:");
       fprintf(ara,"\nPE..... %12.2f",PR[1]);
       fprintf(ara,"\nClear PET..... %12.2f",PR[2]);
       fprintf(ara,"\nColored PET..... %12.2f",PR[3]);
       fprintf(ara,"\nClear glass..... %12.2f",PR[4]);
       fprintf(ara,"\nGreen glass..... %12.2f",PR[5]);
       fprintf(ara,"\nBrown glass..... %12.2f",PR[6]);
       fprintf(ara,"\nAluminum..... %12.2f",PR[7]);
       fprintf(ara,"\nNewspaper..... %12.2f",PR[8]);
       fprintf(ara,"\nMixed PE/PET..... %12.2f",PR[9]);
       fprintf(ara,"\nMixed glass..... %12.2f",PR[10]);
       fprintf(ara,"\nOther material of interest.. %12.2f",PR[11]);
       fprintf(ara,"\nAre you selling (0)-mixed or (1)-sorted PE/PET...
%6.0f",PR[12]);
       fprintf(ara,"\nAre you selling (0)-mixed or (1)-sorted glass....
%6.0f",PR[13]);
       fprintf(ara,"%c",12);
       check_closing(fclose(ara));
       clrscr(); gotoxy(1,1);
       sprintf(ARA,"COPY TEMP.PRT PRN:");
       check_status(system(ARA));
       sleep(DELAY);
       unlink("TEMP.PRT");
     }
   }
   fclose(fil);
   return;
}

void prt_colpoint()
{ int i;
   char ARA[13];
   FILE *ara;
   fil = fopen(D[MODULE][INDEX],"r");
   if ( fil == 0 ) flag_unopened_file(D[MODULE][INDEX]);
   else
   {
     sprintf(ARA,"TEMP.PRT");
```

```
    ara = fopen(ARA,"w");
    if ( ara == 0 ) flag_unopened_file(ARA);
    else
    {
      for(i=1;i<3;i++)
      {
        PT[i]=0.0;
        fgets(B,MFL,fil);
        sscanf(B,"%f",&PT[i]);
      }
      fprintf(ara,"\n  Collection  Point  Data  from  %s  file\n\n\n",
D[MODULE][INDEX] );
      fprintf(ara," Collection Point Data ");
      fprintf(ara,"\nEquipment    cost    per    collection    point..
%12.2f",PT[1]);
      fprintf(ara,"\nOverhead      per      collection      point.......
%12.2f",PT[2]);
      fprintf(ara,"%c",12);
      check_closing(fclose(ara));
      clrscr(); gotoxy(1,1);
      sprintf(ARA,"COPY TEMP.PRT PRN:");
      check_status(system(ARA));
      sleep(DELAY);
      unlink("TEMP.PRT");
    }
  }
  fclose(fil);
  return;
}
```

```
/* FILE:CHECK.C */
#include "variable.h"

void check_rockford_files()
{ /*
     check and make sure that all the files in the ROCKFRD file
     do really exits on disk. BASIC version does not do this!
     You need to call open_rockford before calling this function.
  */
  int i,j,k,m,n;
  FILE *rock;
  frame_screen();
  for(i=1;i<12;i++)
    for(j=1;j<MAX_INDEX;j++)
  {
    if( D[i][j] != NULL )
    {
      gotoxy(1,1);
      printf("\n%c",ULDL);
      for(k=0;k<30;k++)printf("%c",HDL);
      printf("%c\n",URDL);
      if ( STAR[i][j] == 1)
        printf("%c Checking File %12s * %c\n",VDL,D[i][j],VDL);
      else
        printf("%c Checking File %12s   %c\n",VDL,D[i][j],VDL);
      printf("%c",BLDL);
      for(k=0;k<30;k++)printf("%c",HDL);
      printf("%c",BRDL);
      fil = fopen(D[i][j],"rt");
      if ( fil == 0 )
      {
        flag_unopened_file(D[i][j]);
        flag_removing_file(D[i][j]);
        for(m=j;m<(MAX_INDEX-1);m++)
        {
          D[i][m] = D[i][m+1];
          if((m+2) == MAX_INDEX)D[i][m+1] = NULL;
          for(n=1;n<MAX_INDEX;n++) STAR[i][n] =0;
          STAR[i][1] = 1;
          j--;
        };
      }
      else fclose(fil);
      sleep(DELAY);
    };
  };
  rock = fopen(ROCKFORD,"w");
  if ( rock == 0 ) printf("Cannot open ROCKFORD file for update");
  else
  for(i=1;i<12;i++)
    for(j=1;j<MAX_INDEX;j++)
  {
    if ( D[i][j] == NULL ) fprintf(rock,"\n");
    else
```

```
    {
      fprintf(rock,"%s\n",D[i][j]);
    };
  };
  fclose(rock);
  update_star();
  return;
}

void wait_and_go()
{
  int c;
  printf("\nType 1 to exit anything else to continue ");
  c=get_key_1();
  if(c == 1 ) exit(0);
  main_menu();
}

void check_status ( sts )
int sts ;
{
  int c;
  if ( sts == 0 ) return;
  else
  {
    printf("\nError in printer queue");
    printf("\nType 1 to exit anything else to continue ");
    c=get_key_1();
    if(c == 1 ) exit(0);
    main_menu();
  }
  return;
}

void cdbyzero()
{ /* Check for division by 0.0*/
  clrscr();
  if(RM[5]==0.0)VOLP=0.0;
  else
  {
    if(W[18]==0.0)
    {
      printf("The density of recycled plastic can not be equal to
zero\n");
      printf("W[18]==0");
      wait_and_go();
    }
    else VOLP=(RM[5]/W[18])/27.0;
  }
  if(RM[4]==0.0) VOLG=0.0;
  else
  {
    if(W[17]==0.0)
    {
```

```
      printf("The density of recycled glass can not equal zero\n");
      printf("W[17]==0");
      wait_and_go();
    }
    else VOLG=(RM[4]/W[17])/27.0;
  }
  if(RM[3]==0.0) VOLN=0.0;
  else
  {
    if(W[16]==0.0)
    {
      printf("The density of recycled newspaper can equal zero\n");
      printf("W[16]==0");
      wait_and_go();
    }
    else VOLN=(RM[3]/W[16])/27.0;
  }
  if(RM[2]==0.0) VOLA=0.0;
  else
  {
    if(W[15]==0.0)
    {
      printf("The density of recycled aluminum can not equal zero\n");
      printf("W[15]==0");
      wait_and_go();
    }
    else VOLA=(RM[2]/W[15])/27.0;
  }
  if(RM[6]==0.0)VOLO=0.0;
  else
  {
    if(W[19]==0.0)
    {
      printf("The density of recycled Other material can not equal
zero\n");
      printf("W[19]==0");
      wait_and_go();
    }
    else
    {
      VOLO=(RM[6]/W[19])/27.0;
    }
  }
  VOLTOT=VOLP+VOLG+VOLN+VOLA+VOLO;
  if(RM[1]==0.0)
  {
    printf("Whoa baby. You do not have any trash\n\x7");
    printf("RM[1]==0");
    wait_and_go();
  }
  if(W[14]==0.0)
  {
    printf("The average density of solid waste can not equal zero\n");
    printf("\x7 W[14]==0");
```

```
    wait_and_go();
  }
  VOLW=(RM[1]/W[14])/27.0;
  return;
}

void read_star_file ()
{
  int m,k;
  star= fopen("STAR","rt");   /* loads star file      */
  if(star == NULL )
  {
    fclose(star);
    printf("Cannot open STAR file, A new one will be generated\n");
    star= fopen("STAR","w");   /* create one star file      */
    if(star == NULL )
    {
      printf("Cannot open STAR file. Check your directory.\n");
      printf("Possible cause disk full.\n");
      sleep(DELAY);
      exit(0);
    }
    for(m=1;m<12;m++)
      for(k=1;k<MAX_INDEX;k++)
    {
      if ( k == 1 )STAR[m][k]=1;
      else STAR[m][k]=0;
      fprintf(star,"%d\n",STAR[m][k]);
    }
    fclose(star);
  }
  else
  {
    for(m=1;m<12;m++)
      for(k=1;k<MAX_INDEX;k++)
    {
      fgets(B,MFL,star);
      sscanf(B,"%d",&STAR[m][k]);
    }
    fclose(star);
  } ;
  return;
}

void save_current_file()
{ /* Each module calls this file to close the current global file fil */
  int i;
  fclose(fil);
  gotoxy(1,1);
  printf("\n%c",ULDL);
  for(i=0;i<26;i++)printf("%c",HDL);
  printf("%c\n",URDL);
  printf("%c File %12s saved. %c\n",VDL,D[MODULE][INDEX],VDL);
  printf("%c",BLDL);
```

```
  for(i=0;i<26;i++)printf("%c",HDL);
  printf("%c",BRDL);
  return;
}

int get_key_2()
{ /* read the first character from the keyboard and do
     special signaling if it is a special character.
     otherwise read until carriage return is pressed and
     put the result in to the global variable v
  */
  int f,c,i,x,y,j;
  c=getch();
  switch ( c )
  {
    case   8 : /* Back space */
               f = -8 ;
               break;
    case  13 : /* Carriage return */
               f = -13 ;
               break;
    case  27 : /* ESC */
               f = -27;
               break;
    case  32 : /* Space bar */
               f = -32;
               break;
    case  68 : /* d or D to go up one variable */
    case 100 :
               f = -68;
               break;
    case  85 : /* u or U to go up one variable */
    case 117 :
               f = -85;
               break;
    default   : /* read a float number in v and return 0 to indicate
success */
               B[0] = c ;
               x = wherex();
               y = wherey();
               i=1;
               for(;;)
               {
                 B[i]=0;
                 gotoxy(x-1,y);
                 for(j=0;j<(strlen(B)+2);j++)printf(" ");
                 gotoxy(x,y);
                 printf("%s",B);
                 B[i] = getch();
                 if ( B[i] == 13 ) break;
                 if ( B[i] == 27 ) break;
                 if ( B[i] ==  8 )
                 {
                   B[i-1] = 32 ;
```

```
                    B[i] = 0 ;
                    gotoxy(x,y);
                    for(j=0;j<(strlen(B)+2);j++)printf(" ");
                    gotoxy(x,y);
                    printf("%s",B);
                    i--;
                  }
                  else i++;
                  if (i > 19 ) break;
                  if (i < 0 ) i=0;
                }
                if ( sscanf(B,"%f",&v) == 1 ) f = 0;
                else f = -1;
                break;
  }
  return(f);
}

int get_key_1()
{ /*
     read one character form keyboard
     if the character is 0 to 9( 48 to 57 ) then return 0 to 9
     otherwise return the negative ASCII code for further
     processing
  */
  int f,c;
  c=getch();
  switch ( c )
  {
    case  13 : /* Carriage return */
               f = -13;
               break;
    case  27 : /* ESC */
               f = -27;
               break;
    case  32 : /* Space bar */
               f = -32;
               break;
    case  68 : /* d or D for save on disk */
    case 100 :
               f = -68;
               break;
    case  78 : /* n or N for boolean operations */
    case 110 :
               f = -78;
               printf("N");
               break;
    case  80 : /* p or P for print-out */
    case 112 :
               f = -80;
               break;
    case  85 : /* u or U to go up one variable */
    case 117 :
               f = -85;
```

```
                break;
    case  89 : /* y or Y To inquire Yes */
    case 121 :
                f = -89;
                printf("Y");
                break;
    default   : /* read a float number in v and return 0 to indicate
success */
                /* convert ASCII key value to an integer(between 0 & 9) */
                f = c - 48;
                break;
  };
  return(f);
}

void flag_unopened_file( pointer )
char *pointer;
{  /* check for unopened file */
  int c,i,x,y;
  x=23,y=10;
  gotoxy(x,y);
  printf("%c",ULDL);
  for(i=0;i<32;i++)printf("%c",HDL);
  printf("%c",URDL);
  y++;
  gotoxy(x,y);printf("%c Cannot open file  %12s %c",VDL,pointer ,VDL);
  y++;
  gotoxy(x,y);printf("%c Type 1 for main menu           %c",VDL,VDL);
  y++;
  gotoxy(x,y);printf("%c      2 to exit                 %c",VDL,VDL);
  y++;
  gotoxy(x,y);printf("%c      Anything else to continue %c",VDL,VDL);
  y++;
  gotoxy(x,y);printf("%c",BLDL);
  for(i=0;i<32;i++)printf("%c",HDL);
  printf("%c",BRDL);
  c = get_key_1();
  if ( c == 1 ) main_menu();
  if ( c == 2 ) exit(0);
  return;
}

void flag_removing_file( pointer )
char *pointer;
{  /* check for unopened file */
  int i,x,y;
  x=23,y=10;
  gotoxy(x,y);
  printf("%c",ULDL);
  for(i=0;i<32;i++)printf("%c",HDL);
  printf("%c",URDL);
  y++;
  gotoxy(x,y);printf("%c Removing %12s          %c",VDL,pointer ,VDL);
  y++;
```

```
  gotoxy(x,y);printf("%c                                        %c",VDL,pointer
,VDL);
  y++;
  gotoxy(x,y);printf("%c from ROCKFORD file                     %c",VDL,pointer
,VDL);
  y++;
  gotoxy(x,y);printf("%c                                        %c",VDL,pointer
,VDL);
  y++;
  gotoxy(x,y);printf("%c",BLDL);
  for(i=0;i<32;i++)printf("%c",HDL);
  printf("%c",BRDL);
  return;
}

void show_default_file( pointer )
char *pointer;
{  /* check for unopened file */
  int i,x,y;
  x=1,y=1;
  gotoxy(x,y);
  printf("%c",ULDL);
  for(i=0;i<25;i++)printf("%c",HDL);
  printf("%c",URDL);
  y++;
  gotoxy(x,y);printf("%c Working on %12s %c",VDL,pointer ,VDL);
  y++;
  gotoxy(x,y);printf("%c",BLDL);
  for(i=0;i<25;i++)printf("%c",HDL);
  printf("%c",BRDL);
  return;
}
```

```
/* SCREEN.C file */

#include "variable.h"

void output_sec_menu()
{ /* secondary menu for rpt options */
  int c;
  display_jimbo();
  gotoxy(20,15);printf("Available options:");  /*secondary menu (3)*/
  gotoxy(20,16);printf("(1)  View output file");
  gotoxy(20,17);printf("(2)  Print output file");
  gotoxy(20,18);printf("(3)  Destroy an output file");
  gotoxy(20,19);printf("ESCape ... Return to Main Menu");
  gotoxy(20,20);printf("NOTE:  Enter  from  the  keyboard:  1,  2,  3,  or
ESCape");
  c = get_key_1();
  if ( c == -27 ) main_menu();
  switch(c)
  {
    case  1 :
              get_file_name();
              read_rpt();
              screen_1();
              break;
    case  2 :
              get_file_name();
              read_rpt();
              scr_print();
              break;
    case  3 :
              get_file_name();
              delete_rpt();
              update_jimbo();
              break;
    default : output_sec_menu();
              break;
  }
  return;
}

void get_file_name()
{
  int j;
  g o t o x y ( 2 0 , 1 5 ) ; p r i n t f ( "
");
  g o t o x y ( 2 0 , 1 6 ) ; p r i n t f ( "
");
  g o t o x y ( 2 0 , 1 7 ) ; p r i n t f ( "
");
  g o t o x y ( 2 0 , 1 8 ) ; p r i n t f ( "
");
  g o t o x y ( 2 0 , 1 9 ) ; p r i n t f ( "
");
```

```
  g o t o x y ( 2 0 , 2 0 ) ; p r i n t f ( "
");
  gotoxy(5,16);printf("File name ? ");
  gotoxy(18,16);
  for(j=0;j<MFL;j++)B[j]=0;
  gets(B);
  j=file_length_check();
  return;
}

void scr_print()
{
  float COTIP;
  FILE *out;
  int c;
  out=fopen("OUT.TMP","w");
  if(out==0)
  {
    printf("Error opening file OUT.TMP");
    sleep(DELAY);
    return;
  }
  else
  {
    gotoxy(5,19);
    printf("Is printer ready for output report ... (1 Yes/0 No) ");
    c=get_key_1();
    if(c!=1)
    {
      screen_5();
      return;
    }
    else {
    fprintf(out,"\n\n      Report from output file %s\n",B);
    fprintf(out,"      ----------------------------------\n\n\n");
    fprintf(out,"            Revenue, Subsidies, and Credits .....
$/month\n");
    fprintf(out,"\n");
    f p r i n t f ( o u t , "
Plastics.....%7.0f\n",RE[15]);
    f p r i n t f ( o u t , "
Glass.....%7.0f\n",RE[14]);
    f p r i n t f ( o u t , "
Aluminum.....%7.0f\n",RE[7]);
    f p r i n t f ( o u t , "
Newspaper.....%7.0f\n",RE[8]);
    fprintf(out,"                                        Other
Material.....%7.0f\n",RE[11]);
    fprintf(out,"                              Tipping Fee Credit (wt
based).....%7.0f\n",DI[2]);
    fprintf(out,"                                  Government
Subsidies.....%7.0f\n",DI[3]);
    f p r i n t f ( o u t , "
----------\n");
```

```
    fprintf(out,"                                                    Total
Revenue.....................................%7.0f\n",RE[12]);
    fprintf(out,"\n\n");
    fprintf(out,"                              Tipping  Fee  Credit  (volume
based)..%7.0f\n",DI[4]);
    fprintf(out,"\n");
    fprintf(out,"                                                  Adjusted
Total.....................................%7.0f\n",CR[9]);
    fprintf(out,"          (not used in subsequent calculations)");
    fprintf(out,"\n\n\n");
    fprintf(out,"      Recycling Costs ... $/month");
    fprintf(out,"\n");
    fprintf(out,"                                                Collection
Point.............%7.0f\n",CP[4]);
    f    p    r    i    n    t    f    (    o    u    t    ,    "
Collection.....................%7.0f\n",CR[1]);
    fprintf(out,"                                                  Material
Handling.............%7.0f\n",CR[3]);
    f    p    r    i    n    t    f    (    o    u    t    ,    "
Shipping.....................%7.0f\n",SH[7]);
    fprintf(out,"                                                  Plastics
Processing...........%7.0f\n",PL[2]);
    f    p    r    i    n    t    f    (    o    u    t    ,    "
-----------------------------------------------------\n");
    fprintf(out,"                                          Total  Recycling
Costs.....................%7.0f\n",CR[6]);
    fprintf(out,"\n\n");
    fprintf(out,"      Nonrecyclable Costs ... $/month");
    fprintf(out,"\n");
    f    p    r    i    n    t    f    (    o    u    t    ,    "
Collection.....................%7.0f\n",CR[2]);
    fprintf(out,"                                                  Material
Handling.............%7.0f\n",CR[4]);
    f    p    r    i    n    t    f    (    o    u    t    ,    "
Shipping.....................%7.0f\n",SH[6]);
    fprintf(out,"                                                  Tipping
Fees.................%7.0f\n",DI[1]);
    f    p    r    i    n    t    f    (    o    u    t    ,    "
-----------------------------------------------------\n");
    fprintf(out,"                                       Total  Nonrecyclable
Cost.................%7.0f\n",CR[7]);
    fprintf(out,"\n");
    fprintf(out,"                                  Net     Revenue     ...
$/month.............................%7.0f\n",CR[8]);
    fprintf(out,"\n");
    COTIP=-(CR[8]-DI[2])/RM[15]*2000;
    fprintf(out,"        Break Even Tipping Fee...$/ton (Net Revenue =
$0.00)%7.0f\n",COTIP);
    fprintf(out,"\n\n");
    fprintf(out,"                                       Costs Breakdown ...
$/month\n");
    fprintf(out,"\n");
    fprintf(out,"                                Plastic    Glass Aluminum
Newspaper   Other\n");
```

```
    fprintf(out,"       Collection Point  %7.0f  %7.0f  %7.0f      %7.0f
%7.0f\n",NC[ 8],NC[ 9],NC[10],NC[11],NC[12]);
    fprintf(out,"       Collection System %7.0f  %7.0f  %7.0f      %7.0f
%7.0f\n",NC[13],NC[14],NC[15],NC[16],NC[17]);
    fprintf(out,"       Material Handling %7.0f  %7.0f  %7.0f      %7.0f
%7.0f\n",NC[18],NC[19],NC[20],NC[21],NC[22]);
    fprintf(out,"       Shipping          %7.0f  %7.0f  %7.0f      %7.0f
%7.0f\n",SH[ 4],SH[ 2],SH[ 1],SH[ 3],SH[ 5]);
    fprintf(out,"       Processing         %7.0f          0          0
0       0\n",PL[2]);
    f    p    r    i    n    t    f    (    o    u    t    ,    "
-------------------------------------------------------------\n");
    fprintf(out,"       Total             %7.0f  %7.0f  %7.0f      %7.0f
%7.0f\n\n",NC[23],NC[24],NC[25],NC[26],NC[27]);
    fprintf(out,"                                 Plastic   Collection   Cost/lb
%7.2f\n\n",NC[28]);
    f    p    r    i    n    t    f    (    o    u    t    ,    "
Wt.(lbs) Vol.(yd^3)\n");
    fprintf(out,"       Total Waste Stream per month                  %7.0f
%7.0f\n\n",RM[1],VOLW);
    fprintf(out,"                  Recycled Plastic ..................  %7.0f
%7.0f\n",RM[5],VOLP);
    fprintf(out,"                  Recycled Glass ....................  %7.0f
%7.0f\n",RM[4],VOLG);
    fprintf(out,"                  Recycled Newspaper ................  %7.0f
%7.0f\n",RM[3],VOLN);
    fprintf(out,"                  Recycled Aluminum .................  %7.0f
%7.0f\n",RM[2],VOLA);
    fprintf(out,"                  Other Recycled Material ...........  %7.0f
%7.0f\n",RM[6],VOLO);
    f    p    r    i    n    t    f    (    o    u    t    ,    "
-------------------------------------------------------------\n");
    fprintf(out,"       Total recyclables removed from waste stream %7.0f
%7.0f\n",RM[15],VOLTOT);
    fprintf(out,"                              %%   of   total   waste   stream
%3.0f%%        %3.0f%%\n",RM[15]*100/RM[1],VOLTOT*100/VOLW);
    fprintf(out,"%c",12);
    fclose(out);
    /* Use the system print queue to spool the temporary file */
    clrscr();
    gotoxy(1,1);
    system("COPY OUT.TMP PRN:");
    sleep(DELAY);
    }
  }
  return;
}

void update_jimbo()
{  /* Open, re-write and close the JIMBO file */
  int g,c;
  clrscr();
  frame_screen();
  jim = fopen("JIMBO","w");
```

```
  if(jim==0)
  {
    printf("\n***Can't find JIMBO file***");
    sleep(DELAY);
    return;
  }
  else
  {
    for(g=1;g<31;g++)
    {
      if( F[g]!=NULL) fprintf(jim,"%s\n",F[g]);
      JIMBO_INDEX=g;
    };
    check_closing (fclose(jim));
    if( JIMBO_INDEX > 31 )
    {
      gotoxy(5,20);printf("You reached the maximum number of report
files.");
      gotoxy(5,23);printf("Hit the space bar to return to screen 5.");
      c = get_key_1();
      if(c==-32) screen_5();
    };
  };
  return;
}

void display_jimbo()
{ /* Read the JIMBO file and display the existing *.rpt files */
  int i,c;
  clrscr();
  frame_screen();
  JIMBO_INDEX=0;
  for(i=1;i<31;i++) free(F[i]);
  jim = fopen("JIMBO","r");
  if(jim==0)
  {
    gotoxy(2,2);
    printf("Can't find JIMBO file");
    gotoxy(2,3);
    printf("Creating a new one !");
    jim = fopen("JIMBO","w");
    for(i=1;i<31;i++)fprintf(jim,"%s\n",F[i]);
    check_closing(fclose(jim));
    JIMBO_INDEX=1;
    sleep(DELAY);
  }
  else
  {
    gotoxy(5,2);printf("Existing output files:");
    for(i=1;i<31;i++)   /*lists output file names*/
    {
      for(c=0;c<MFL;c++)B[c]=0;
      if(fgets(B,MFL,jim)==NULL)
      {
```

```
        JIMBO_INDEX=i;
        goto Found;
      }
      else
      {
        c = file_length_check();
        c = strlen(B);
        if ( c == 0 )
        {
          JIMBO_INDEX = i ;
          goto Found;
        }
        else
        {
          F[i]=malloc(12);  /* allocate memory for the string */
          strcpy(F[i],B);
          if(i<11)          /*makes in three columns*/
          {
            gotoxy(15,3+i);
            printf("%s",F[i]);
          }
          else if( i<21 )
          {
            gotoxy(35,i-7);
            printf("%s",F[i]);
          }
          else
          {
            gotoxy(55,i-17);
            printf("%s",F[i]);
          };
        };
      };
    };
  };
Found:
  check_closing(fclose(jim));
  if(JIMBO_INDEX>30)
  {
    gotoxy(5,20);printf("You reached the maximum number of print-out
files.");
    gotoxy(5,23);printf("Hit the space bar to return to screen 5.");
    c = get_key_1();
    if(c==-32) screen_5();
  };
  return;
}

void read_rpt()
{ /* Read the report files */
  int i;
  fil = fopen(B,"r");
  if(fil==0) flag_unopened_file(B);
  else
```

```
    {
      fscanf(fil,"%f",&RE[15]);
      fscanf(fil,"%f",&RE[14]);
      fscanf(fil,"%f",&RE[7]);
      fscanf(fil,"%f",&RE[8]);
      fscanf(fil,"%f",&RE[11]);
      fscanf(fil,"%f",&DI[2]);
      fscanf(fil,"%f",&DI[3]);
      fscanf(fil,"%f",&RE[12]);
      fscanf(fil,"%f",&DI[4]);
      fscanf(fil,"%f",&CR[9]);
      fscanf(fil,"%f",&CP[4]);
      fscanf(fil,"%f",&CR[1]);
      fscanf(fil,"%f",&CR[2]);
      fscanf(fil,"%f",&SH[7]);
      fscanf(fil,"%f",&PL[2]);
      fscanf(fil,"%f",&CR[6]);
      fscanf(fil,"%f",&CR[2]);
      fscanf(fil,"%f",&CR[4]);
      fscanf(fil,"%f",&SH[6]);
      fscanf(fil,"%f",&DI[1]);
      fscanf(fil,"%f",&CR[7]);
      fscanf(fil,"%f",&CR[8]);
      fscanf(fil,"%f",&RM[1]);
      fscanf(fil,"%f",&RM[5]);
      fscanf(fil,"%f",&RM[4]);
      fscanf(fil,"%f",&RM[3]);
      fscanf(fil,"%f",&RM[2]);
      fscanf(fil,"%f",&RM[6]);
      fscanf(fil,"%f",&RM[15]);
      fscanf(fil,"%f",&VOLW);
      fscanf(fil,"%f",&VOLP);
      fscanf(fil,"%f",&VOLG);
      fscanf(fil,"%f",&VOLN);
      fscanf(fil,"%f",&VOLA);
      fscanf(fil,"%f",&VOLO);
      fscanf(fil,"%f",&VOLTOT);
      for(i=1;i<29;i++) fscanf(fil,"%f",&NC[i]);
      fscanf(fil,"%f",&SH[4]);
      fscanf(fil,"%f",&SH[2]);
      fscanf(fil,"%f",&SH[1]);
      fscanf(fil,"%f",&SH[3]);
      fscanf(fil,"%f",&SH[5]);
      fscanf(fil,"%f",&PL[2]);
      fclose(fil);
    }
    return;
}

void delete_rpt()
{ /* To delete a file from the current directory and mark it
     on JIMBO file
   */
   int i,c,OLD;
```

```
  char a[20];
  OLD=0;
  for(i=1;i<31;i++)
  {
    if(strcmp(B,F[i])==0) OLD = i ;
    /* Now do some garbage compaction */
    if(OLD > 0 )
    {
      F[i]=F[i+1];
      F[i+1]=0;
    };
  };
  if( OLD == 0 )
  {
    gotoxy(34,19);printf("Cannot find file %s",B);
    gotoxy(34,20);printf("Type Esc to continue");
    c=get_key_1();
    if(c==-27)main_menu();
    else return;
  };
  fil = fopen(B,"r");
  if( fil == 0 ) flag_unopened_file( B );
  else fclose(fil);
  sprintf(a,"DEL %s",B);
  system(a);
  return;
}

int file_length_check()
{
  int j,h;
  h = strlen(B);
  if ( h > 13 )
  {
    printf("\n* * File name %s is too long!\n",B);
    h=-1;
  }
  else if ( h <  0 )
  {
    printf("\n* * File name %s is too short!\n",B);
    h=-1;
  }
  else
  { /* Make sure that the file name is a valid one */
    for(j=0;j<MFL;j++)
      if((isalnum(B[j])==0)&&(B[j]!='.')&&(B[j]!='_'))
      { /* Unvalid filename construction */
        B[j]=0;
      }
      else B[j]=toupper(B[j]);
  }
  return(h);
}
```

```
void save_rpt()
{
  int i,c;
  fil = fopen(B,"w");
  if( fil == 0 )
  {
    printf("\nsave_rpt");
    flag_unopened_file( B );
  }
  else
  { /* check jimbo array for matching name */
    for(i=1;i<31;i++)
    {
      if(F[i]==NULL)
      {
        JIMBO_INDEX=i;
        goto Found;
      }
      else if(strcmp(F[i],B)==0)
      {
        JIMBO_INDEX=i;
        gotoxy(5,17);
        printf("File %s exists, overwrite? (0 is No,Default is Yes)",B);
        c = get_key_1();
        if ( c == 0 )
        {
        gotoxy(1,2);
        printf("                                   ");
        main_menu();
        }
        else  goto Found;
      };
    };
  };
Found:
  F[JIMBO_INDEX] = malloc(strlen(B));
  strcpy(F[JIMBO_INDEX],B);
  /*Data Screen #1*/
  fprintf(fil,"%f\n",RE[15]);
  fprintf(fil,"%f\n",RE[14]);
  fprintf(fil,"%f\n",RE[7]);
  fprintf(fil,"%f\n",RE[8]);
  fprintf(fil,"%f\n",RE[11]);
  fprintf(fil,"%f\n",DI[2]);
  fprintf(fil,"%f\n",DI[3]);
  fprintf(fil,"%f\n",RE[12]);
  fprintf(fil,"%f\n",DI[4]);
  fprintf(fil,"%f\n",CR[9]);
  /*Data Screen #2*/
  fprintf(fil,"%f\n",CP[4]);
  fprintf(fil,"%f\n",CR[1]);
  fprintf(fil,"%f\n",CR[2]);
  fprintf(fil,"%f\n",SH[7]);
  fprintf(fil,"%f\n",PL[2]);
```

```
    fprintf(fil,"%f\n",CR[6]);
    fprintf(fil,"%f\n",CR[2]);
    fprintf(fil,"%f\n",CR[4]);
    fprintf(fil,"%f\n",SH[6]);
    fprintf(fil,"%f\n",DI[1]);
    fprintf(fil,"%f\n",CR[7]);
    fprintf(fil,"%f\n",CR[8]);
    /*Data Screen #4*/
    fprintf(fil,"%f\n",RM[1]);
    fprintf(fil,"%f\n",RM[5]);
    fprintf(fil,"%f\n",RM[4]);
    fprintf(fil,"%f\n",RM[3]);
    fprintf(fil,"%f\n",RM[2]);
    fprintf(fil,"%f\n",RM[6]);
    fprintf(fil,"%f\n",RM[15]);
    fprintf(fil,"%f\n",VOLW);
    fprintf(fil,"%f\n",VOLP);
    fprintf(fil,"%f\n",VOLG);
    fprintf(fil,"%f\n",VOLN);
    fprintf(fil,"%f\n",VOLA);
    fprintf(fil,"%f\n",VOLO);
    fprintf(fil,"%f\n",VOLTOT);
    /* Data Screen #5 */
    for(i=1;i<29;i++) fprintf(fil,"%f\n",NC[i]);
    fprintf(fil,"%f\n",SH[4]);
    fprintf(fil,"%f\n",SH[2]);
    fprintf(fil,"%f\n",SH[1]);
    fprintf(fil,"%f\n",SH[3]);
    fprintf(fil,"%f\n",SH[5]);
    fprintf(fil,"%f\n",PL[2]);
    fclose(fil);
    gotoxy(5,18);
    printf("Saving Of Output Data File %s Complete",B);
    sleep(DELAY);
    return;
}
```

Appendix B

DOCUMENTATION

How to compile one or more file(s) of *.C ?

TURBO C is able to handle separate compilation of multiple source files. When we build a program from more than one C source file, however, we have to tell TURBO C exactly which files are involved. That means we have to create a project file. Creating a project file is as simple as listing the names of our C source files. The project file will simply contain the lines naming the files to be compiled and linked. We do not to type the .C extensions. We have to type variable.h file in our in our case such as "pride.c (variable.h)". TURBO C assumes any file without an extension is a .C file. The order of the files is not important.

pride.c (variable.h)

compute.c (variable.h)

output.c (variable.h)

input.c (variable.h)

hardcopy.c (variable.h)

check.c (variable.h)

readdata.c (variable.h)

screen.c (variable.h)

(Each of them is to be dependent on variable.h).

Now we can save our file as PRIDE.PRJ (selecting WRITE TO from the FILE menu). The name of the project file (PRIDE.PRJ) is not the same as the name of the main file (PRIDE.C). The two names could have been the same (but not the extensions), but they do not have to be. In our case the executable file will be PRIDE.EXE.

Now that we have a project file, all we need to do is to tell TURBO C what project we want to make. This is done by entering the name

of project file on the PROJECT menu (by pressing Alt-P to get the Project Menu and choose Project name). Once our project name is entered, we can simply press F9 to MAKE the executable file. To RUN this program, we have to press Ctrl-F9.

In TURBO C the make cycle stops :

- o once an executable file has been produced,
- o to report some type of error,
- o by pressing Ctrl-Break,
- o when the compiler generates messages.

The execution of PRIDE needs the following files in a directory or on a floppy :

pride.prj, variable.h, pride.c, input.c, output.c, hardcopy.c, readdata.c, check.c, compute.c, screen.c, command.com, stdio.h, conio.h, dos.h, io.h, string.h, fcntl.h, sys\stat.h, ctype.h, stdlib.h, tc.exe.

If STAR, JIMBO and ROCKFORD files do not exist, they will be created during the execution of PRIDE.EXE.

Appendix C

MODEL DESCRIPTION

This appendix describes the model's modular structure, the calculations that each module performs, and the data sets that it requires. Figure 1 shows the interrelations between the different modules and data sets.

At the top of Figure 1 the regional and policy data sets are shown. These data sets contain : regional information, incentive information, and some other collection policy information. This information allows the first modules to calculate the physical dimensions which the rest of the modules must consider, number of collection points per month, total waste stream etc...

Shown in the center of the figure 1 surrounded by a dotted line are modules which perform the calculations for the different stages of the model. The Raw Materials module calculates the total quantities of each of the different waste materials. The Collection Point module calculates the costs associated with the collection point, not including the actual collection. So that the costs associated with bags or buckets, reverse vending equipment, or public awareness programs are all considered in the Collection Point module.

The Collection System module considers the costs of actually collecting the raw materials from the collection points. So the costs of collection equipment, associated labor, and mileage cost are included.

The Material Handling module considers the costs of handling the raw materials after they have been collected. Functions such as sorting, granulation, shredding, compacting are considered in the Material Handling module so long as those functions are performed outside as part

of the actual collection then they are included in the collection in the collection module.

The Shipping module considers the costs of transporting the raw materials to their final destinations which may include : landfills, plastics processing plants, or processing centers for other recyclable materials. The Disposal module considers the cost of disposing materials in landfills.This includes Government subsidies and credits for landfill relief based on both weight and volume. The Plastics Process module considers the costs associated with processing the different plastic material. The Recyclable Market considers the value of all of the recyclables collected.

In Figure 1, to the right of the central modules (outside the dotted line box), are the remainder of the data sets. Most of the information contained in the data sets is cost information in the Solid Waste Characteristics data set is information which is unlikely to be varied in analysis. It includes items such as the average amount of waste created per capita per month and the average densities of the materials in the solid waste stream.

he Collection Equipment, Labor, and Handling Equipment data sets contain information which characterize different collection systems and their costs. The analysis of different collection systems is accomplished by using different Collection Equipment, Labor, and Handling Equipment dat sets. The shipping data set contains shipping factors such as distance, capacity, and cost per mile. The disposal data set contains tipping fees and government subsidies. The Plastics Processing data set includes equipment costs, process costs and efficiency parameters. The price data set contains a price schedule for

the different recyclable materials, including prices for sorted and unsorted glass and plastics.

to the left of the central modules in the figure are the Cost and Revenue modules. These modules accumulate the costs and revenues. These modules perform calculations such as a per material cost of recycling breakdown, including allocating costs to different recyclable material types. Most of the output information is tabulated in these modules.

MODEL

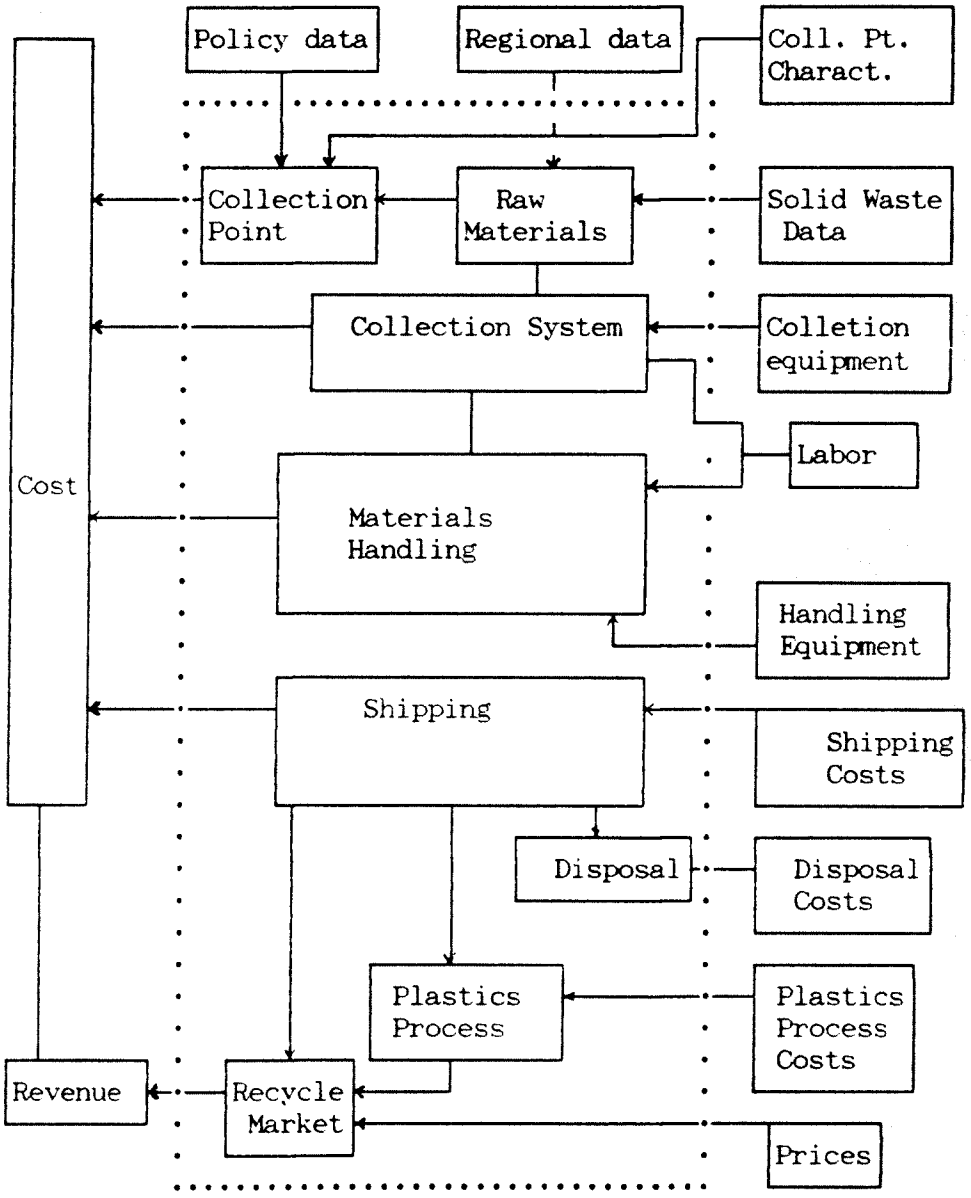

Figure 1.

Appendix D

PROGRAM VARIABLES

RAW MATERIAL

```
Total Solid Waste/month = Avg Quantity Solid Waste Per Capita /
                          month * Population Density * Square Miles

Recyclable Aluminum/month = Total Solid Waste / month *
                            Aluminum Compliance Rate *
                            Percent Solid Waste that is Aluminum

Recyclable Newspaper/month = Total Solid Waste / month *
                             Newspaper Compliance Rate *
                             Percent Solid Waste that is Newspaper

Recyclable Glass/month = Total Solid Waste / month *
                         Glass Compliance Rate *
                         Percent Glass Solid Waste

Recyclable Plastics/month = Total Solid Waste / month *
                            Plastics Compliance Rate *
                            Percent Plastic Solid Waste

Recyclable Other Material/month = Total Solid Waste / month *
                                  Other Material Compliance rate *
                                  % Solid Waste that is Other Material

Recyclable PE/month = Recyclable Plastic/month *
                      Percent Plastics that are PE

Recyclable PET/month = Recyclable Plastic/month *
                       Percent PET that is PET

Recyclable clear PET/month = Recyclable PET *
                             percent PET that is clear

Recyclable colored PET/month = Recyclable PET *
                               Percent PET that is colored

Recyclable Clear Glass/month = Recyclable Glass *
                               Percent Glass that is clear

Recyclable Green Glass/month = Recyclable Glass *
                               Percent Glass that is Green

Recyclable Brown Glass/month = Recyclable Glass *
                               Percent Glass that is Brown

Solid Waste Displ. (wt) = Recyclable Plastics / month *
                          Solid Waste Density / Plastic Dens. +
                          Recyclable Aluminum * Solid Waste Dens. /
                          Aluminum Density + Recyclable Glass *
                          Solid Waste Density / Glass Density +
                          Recyclable Newspaper * Solid Waste Density /
```

Newspaper Density +
Recyclable Other material *
Solid Waste Density /
Other Material Density

Recyclable Solid Waste/month = Recyclable Aluminum +
Recyclable Newspaper +
Recyclable Glass +
Recyclable Plastics +
Recyclable Other Material

Post Recycled Solid Waste/month = Total Solid Waste / month -
Recyclable Solid Waste / month

Recyclable Other Material Vol. = Recyclable Other Material /
Other material Density

Recyclable Aluminum Vol. = Recyclable Aluminum / Aluminum Density

Recyclable Glass Vol. = Recyclable Glass / Glass Density

Recyclable Plastics Vol. = Recyclable Plastics / Plastics Density

Recyclable Newspaper Vol. = Recyclable Newspaper / Newspaper Density

Recyclable Vol. = Recyclable Other Material Vol. +
Recyclable Aluminum Vol. +
Recyclable Glass Vol. + Recyclable Plastics +
Recyclable Newspaper Vol.

Total Solid Waste Vol. = Total Solid Waste / Solid Waste Density

COLLECTION POINT

Payout/month = Recyclable Plastics * Payout Rate

Collection Point Equip Cost/month = Equip Cost *
Number of collection points

Overhead/month = Overhead/unit * Number of collection points

Total Col. Point Cost/month = Pay-out/month +
Public Awareness Program Cost/month +
Collection Point Equip Cost/month +
Overhead/month

COLLECTION SYSTEM

Avg Miles/Collection Point = Total Route Miles/Total Collection Points

Labor Cost/month = Collector cost / month / unit *
Σ[number of units * staff per unit]

Equip Cost/month = Equip cost/month/unit * number of units

Overhead Cost/month = Overhead Cost/month/unit * number of units

Mileage Cost/month = Avg miles/collection point *
Avg number collection points/unit *
cost/mile/unit

Total Cost of Collection = Labor Cost / month + Equip Cost / month +
Overhead Cost + Mileage Cost / month

Plastic Collection System Cost = Recyclable Collection Cost *
Recyclable Plastics Vol /
Recyclable Vol

Glass Collection System Cost = Recyclable Collection Cost *
Recyclable Glass Vol / Recyclable Vol

Aluminum Collection System Cost = Recyclable Collection Cost *
Recyclable Aluminum Vol. /
Recyclable Vol.

Newspaper Collection System Cost = Recyclable Collection Cost *
Recyclable Newspaper Vol. /
Recyclable Vol.

Other Material Col. System Cost = Recyclable Collection Cost *
Recyclable Other Material Vol. /
Recyclable Vol.

MATERIALS HANDLING

Total Number of Operators = Σ [# of Operators across all functions]

Materials Handling Labor Cost/month = Material handler cost/month *
Total Number of Operators

Material Handling Equip Cost/month = Σ [Total Number of Machines *
equip cost/unit/month across
all functions]

Total Materials Handling Cost = Materials Handling Equip Cost +
Materials Handling Labor Cost / month +
Materials Handling Overhead / month

Plastics Material Handling Cost = (Granulation Operators *
Material Handling Labor Rate) +
(Granulation Machines *
Granulation Equipment cost) +
(Baling Operators *
Material Handling Labor Rate) +
Baling Machines *
Baling Equipment Cost) +
(Sorting Operators *
Material Handling Labor Rate *
Recyclable Plastics Vol.) /

(Recyclable Plastics Vol. +
Recyclable Glass Vol. +
Recyclable Aluminum Vol.) +
(Overhead * Recyclable Plastics Vol.)

Recyclable Vol.

Glass Material Handling Cost = (Crusher Operators *
Material Handling Labor Rate *
Recyclable Glass Vol.) /
(Recyclable Glass Vol.+
Recyclable Aluminum) +
(Crushing Machines *
Crushing Equipment Cost *
Recyclable Glass Vol.) /
(Recyclable Glass Vol.+
Recyclable Aluminum Vol.) +
(Sorting Operators *
Material Handling Labor Rate *
Recyclable Glass Vol.) /
(Recyclable Plastics Vol. +
Recyclable Glass Vol. +
Recyclable Aluminum Vol.) +
(Overhead * Recyclable Glass Vol.) /
Recyclable Vol.

Aluminum Material Handling Cost = (Crusher Operators *
Material Handling Labor Rate *
Recyclable Aluminum Vol.) /
(Recyclable Glass Vol. +
Recyclable Aluminum Vol.) +
(Crushing Machines *
Crushing Equipment Cost *
Recyclable Aluminum Vol.) /
(Recyclable Glass Vol. +
Recyclable Aluminum Vol.) +
(Sorting Operators *
Material Handling Labor Rate *
Recyclable Aluminum Vol.) /
(Recyclable Plastics Vol. +
Recyclable Glass Vol. +
Recyclable Aluminum Vol.) +
(Overhead * Recyclable Aluminum Vol.)

Recyclable Vol.

Newspaper Material Handling Cost = Recycling Newspaper Vol. *
Overhead/Recyclable Vol.

Other Material Handling Cost = Recycling Other Material Vol. *
Overhead / Recyclable Vol.

PLASTICS PROCESSING

Plastics Processing Labor Cost/month = Number of Operators *

Material Handler Cost/month

Plastics Processing Direct Process Cost = Recyclable Plastics *
Direct Process Cost

Total Plastics Proc. Cost/month = Plastic Processing Labor Cost / month
+
Plastics Processing Overhead / month +
Plastics Processing Equip Cost / month
+
Plastics Processing Direct Process
Cost

SHIPPING

Shipping Cost for Aluminum/month = Recyclable Aluminum / month *
Shipment Capacity for Aluminum *
Aluminum Market Distance *
Aluminum Cost / mile

Shipping Cost for Glass/month = (Recyclable Glass/month /
Shipment Capacity for Glass) *
Glass Market Distance * Glass
Cost/mile

Shipping cost for Newspaper/month = (Recyclable Newspaper/month /
Shipment Capacity for newspaper) *
Newspaper market Distance *
Newspaper Cost / mile

Shipping Cost for Plastics/month = (Recyclable Plastics / month /
Shipment Capacity for Plastics) *
Plastics Market Distance *
Plastics Cost / mile

Shipping Cost for Other Material/month = (Recyclable Other
Material/month /
Shipment Capacity for Other Material
) *
Other Material market Distance *
Other Material Cost / mile

Shipping Cost For Solid Waste = (Post Recycling Solid Waste /
Shipment Capacity for Solid Waste) *
Solid Waste Market Distance *
Solid Waste Cost / mile

Total Recyclable Shipping Cost/month = Ship. Cost for Aluminum/month +
Ship. Cost for Glass/month +
Ship. Cost for Newspaper/month +
Ship. Cost for Plastics/month +
Ship. Cost for Other
Material/month

DISPOSAL

```
Total Tipping Fees = Post Recycling Solid Waste / 2000lbs *
                     Tipping Fee * Simultaneous Collect

Tipping Fee Credit (wt) = Recycled Solid Waste/2000lbs *
                          Tipping Fee * (1 - Simultaneous Collect)

Gov Credit = Recycled Solid Waste/2000 * Gov Abatement

Tipping Fee Credit (vol) = ( Solid Waste Density /
                             Plastic Density *
                             Recyclable Plastic / 2000 +
                             Solid Waste Density /
                             Aluminum Density *
                             Recyclable Aluminum / 2000 +
                             Solid Waste Density /
                             Glass Density *
                             Recyclable Glass / 2000 +
                             Solid Waste Density /
                             Newspaper Density *
                             Recyclable Newspaper / 2000 +
                             Solid Waste Density /
                             Other Material Density *
                             Recyclable Other Material / 2000 ) *
                             Tipping Fee
```

RECYCLABLES MARKET

```
PE Revenue/month = Process Efficiency * Price/lb PE * Recyclable PE

Clear PET Revenue/month = Process Efficiency * Price/lb Clear PET *
                          Recyclable Clear PET

Colored PET Revenue/month = Process Efficiency * Price/lb Colored PET *
                            Recyclable Colored PET

Clear Glass Revenue/month = Price/lb Clear Glass * Recyclable Clear
Glass

Brown Glass Revenue/month = Price/lb Brown Glass * Recyclable Brown
Glass

Green Glass Revenue/month = Price/lb Green Glass * Recyclable Green
Glass

Aluminum Revenue/month = Price/lb Aluminum * Recyclable Aluminum

Newspaper Revenue/month = Price/lb Newspaper * Recyclable Newspaper

Mixed PE/PET Revenue/month = Process Efficiency * Price/lb Mixed PE/PET
*
                             Recyclable Mixed PE/PE
```

```
Mixed Glass Revenue/month = Price/lb Mixed Glass * Recyclable Mixed
Glass

Other Material Revenue/month = Price/lb Other Material *
                               Recyclable Other Material

Total Revenue = PE Revenue/month + Clear PET Revenue/month +
                Colored PET Revenue/month + Clear Glass Revenue/month +
                Brown Glass Revenue/month + Green Glass Revenue/month +
                Aluminum Revenue/month + Newspaper Revenue/month +
                Mixed PE/PET Revenue/month + Mixed Glass Revenue/month +
                Other Material Revenue/month + Tipping Fee Credit (wt) +
                gov. Subsidy

Post Recycled Solid Waste/month = Total Solid Waste/month -
                                  Recycled /Solid Waste/month

Total Glass Rev = Clear Glass Revenue + Brown Glass Revenue +
                  Green Glass Revenue + Mixed Glass Revenue

Total Plastic Rev = PE Revenue + Clear PET Revenue + Colored PET Revenue
+
                    Mixed PE/PET Revenue


COST and REVENUE

Recyclable Collection Cost/month = (Total Collection Cost *
                                   Recycled Solid Waste/Total Solid
Waste *
                                   Simultaneous Collect) +
                                   ((1 - Simultaneous Collect) *
                                   Total Collection Cost)

Non Recyclable Collection Cost/month = ( Total Collection Cost -
                                         Recyclable Cost ) *
                                         Simultaneous Collection

Recyclable Material Handling Cost = (((Material Handling Labor Cost +
                                    Material Handling Overhead ) *
                                    ( Recyclable Solid Waste /
                                    Total Solid waste ) +
                                    Material Handling Equipment cost ) *
                                    Simultaneous Collect ) +
                                    ( Total Material Handling Cost *
                                    (1 - Simultaneous Collect))

NonRecycling Material Handling Cost = (Total Material Handling Cost -
                                      Recycling Material Handling Cost )
*
                                      Simultaneous Collect

Total Recycling Revenue = Total Plastics Revenue +
                          Total Glass Revenue +
```

```
                                Total Aluminum Revenue +
                                Total Newspaper Revenue +
                                Total Other Material Revenue +
                                Tipping Fee Credit + Gov Subsidy

Total Recycling Cost = Total Collection Point Cost +
                       Recycling Collection Cost +
                       Recyclable Material Handling Cost +
                       Total Recycling Shipping Cost +
                       Total Plastics Processing Co.

Total NonRecycling Cost = Total NonRecycling Collection Cost +
                          Shipping Cost for Solid Waste +
                          Total NonRecycling Material Handling Cost +
                          Total Tipping Fees

Net Revenue = Total Revenue - Total Recycling Cost-Total NonRecycling
Cost

Adjusted Total = Total Revenue - Tipping Fee Credit (wt) +
                 Tipping Fee Credit (vol)

Collect Pt. Plastic = Playout/month + Collect Pt. Equipment Cost +
                      ( Program Cost * ( Recyclable Plastic Vol. /
                      Recyclable Vol. )) + (Collect Pt. Overhead *
                      Recyclable Plastic Vol. ) /
                      ( Recyclable Plastics Vol. +
                      Recyclable Glass Vol. + Recyclable Aluminum Vol )

Collect Pt. Glass = (Program Cost * Recyclable Glass Vol. /
                    Recyclable Vol. ) + ( Collect Pt. Overhead *
                    Recyclable Glass Vol. ) / ( Recyclable Plastics Vol.
+
                    Recyclable Glass Vol. + Recyclable Aluminum Vol. )

Collect Pt. Aluminum = ( Program Cost * Recyclable Aluminum Vol. /
                       Recyclable Vol.) + ( Collect Pt. Overhead *
                       Recyclable Aluminum Vol.) /
                       ( Recyclable Plastics Vol. +
                       Recyclable Glass Vol. + Recyclable Aluminum Vol.)

Collect Pt. Newspaper = Program Cost * Recyclable Newspaper Vol. /
                        Recyclable Vol.

Collect Pt. Other Material = Program Cost * Recyclable Other Material
Vol./
                                Recyclable Vol.

Total Plastic Cost = Collect Pt. Plastic +
                     Plastics Collection Sys. Cost +
                     Plastics Material Handling Cost +
                     Plastics Shipping Cost + Plastics Processing Cost

Total Glass Cost = Collect Pt. Glass + Glass Collection Sys. Cost +
```

Glass Material Handling Cost + Glass Shipping Cost

Total Aluminum Cost = Collect Pt. Aluminum +
Aluminum Collection Sys. Cost +
Aluminum Material Handling +
Aluminum Shipping Cost

Total Newspaper Cost = Collect Pt. Newspaper +
Newspaper Collection Sys. Cost +
Newspaper Material Handling +
Newspaper Shipping Cost

Total Other Material Cost = Collect Pt Other Material +
Other Material Collection Sys Cost +
Other Material Handling Cost +
Other Material Shipping Cost

Plastic Recycling Cost per lb = Total Plastics Cost / Recyclable Plastic

The following is a list of variable names used in PRIDE's input screens and the corresponding variable name in the source code of PRIDE. This information is presented here to aid in future modifications of the program.

```
Square Miles =                          R[1]
Population Density =                    R[2]
Route Miles =                           R[3]
Number of Collection Points =           R[4]

Per Capita Compliance Rate For Plastics =                          P[1]
Per Capita Compliance Rate For Glass =                             P[2]
Per Capita Compliance Rate For Aluminum =                          P[3]
Per Capita Compliance Rate For Newspaper =                         P[4]
Per Capita Compliance Rate For Other Material of Interest =        P[5]
Pay out Rate =                                                     P[6]
Public Awareness Program Cost =                                    P[7]
Collection Schedule =                                              P[8]
Simultaneously =                                                   P[9]

Collection Point Equipment Cost[$/month] =  PT[1]
Collection Point Overhead[$ / month] =      PT[2]

Average Quantity of Solid Waste Created Per Person Per Month = W[1]
Percentage of Solid Waste that is Aluminum =                   W[2]
Percentage of Solid Waste that is Newspaper =                  W[3]
Percentage of Solid Waste that is Glass =                      W[4]
Percentage of Solid Waste that is Plastics =                   W[5]
Percentage of Solid Waste that is Other Material of Interest = W[6]
Percentage of Plastics that are PE =        W[7]
Percentage of Plastics that are PET =       W[8]
Percentage of PET that is Clear =           W[9]
Percentage of PET that is colored =         W[10]
Percentage of Glass that is Clear =         W[11]
Percentage of Glass that is Green =         W[12]
Percentage of Glass that is Brown =         W[13]
The Average Density of Solid Waste =        W[14]
The Average Density of Uncollapsed
  Plastic pulled from waste stream =        W[15]
The Average Density of Uncollapsed
  Aluminum pulled from waste stream =       W[16]
The Average Density of Uncrushed
  Glass pulled from waste stream =          W[17]
The Average Density of Newspaper =          W[18]
The Average Density of Other Material =     W[19]

# of Units Vehicle Type 1 =                C[1]
Weight Capacity Vehicle Type 1 =            C[2]
Volume Capacity Vehicle type 1 =            C[3]
Equip Cost Vehicle type 1 =                 C[4]
Overhead Vehicle type 1 =                   C[5]
Cost/mile vehicle Type 1 =                  C[6]
Staff/unit Vehicle Type 1 =                 C[7]
Avg # Collect points/month Vehicle Type 1 = C[8]
```

# of Units Vehicle Type 2 = C[9]
Weight Capacity Vehicle Type 2 = C[10]
Volume Capacity Vehicle Type 2 = C[11]
Equip Cost Vehicle Type 2 = C[12]
Overhead vehicle Type 2 = C[13]
Cost/mile Vehicle Type 2 = C[14]
Staff/unit Vehicle Type 2 = C[15]
Avg # Collect points/month Vehicle Type 2 = C[16]
# of Units Vehicle Type 3 = C[17]
Weight Capacity Vehicle type 3 = C[18]
Volume Capacity Vehicle Type 3 = C[19]
Equip Cost Vehicle type 3 = C[20]
Overhead Vehicle Type 3 = C[21]
Cost/mile Vehicle Type 3 = C[22]
Staff/unit Vehicle type 3 = C[23]
Avg # Collect points/month Vehicle Type 3 = C[24]
# of Units Vehicle Type 4 = C[25]
Weight Capacity Vehicle Type 4 = C[26]
Volume Capacity Vehicle type 4 = C[27]
Equip Cost Vehicle Type 4 = C[28]
Overhead Vehicle Type 4 = C[29]
Cost/mile Vehicle type 4 = C[30]
staff/unit Vehicle Type 4 = C[31]
Avg # Collect points/month Vehicle Type 4 = C[32]

Collector Wage = L[1]
Material Handler Wage = L[2]

Sorting-Manual # operators = H[1]
Granulation/shredding # operators = H[2]
Granulation/shredding # machines = H[3]
Granulation/shredding equip cost = H[4]
Baling # operators = H[5]
Baling # machines = H[6]
Baling equip cost = H[7]
Crushing[Alum/Glass] # machines = H[9]
Crushing[Alum/Glass] # operators = H[8]
Crushing[Alum/Glass] equip cost = H[10]
Monthly Overhead = H[11]

Distance from the materials handling facility to Glass Market = S[1]
Distance from the materials handling facility to Newspaper Market = S[2]
Distance from the materials handling facility to Aluminum Market = S[3]
Distance from the materials handling facility to
Plastic Processing Plant = S[4]
Distance from the materials handling facility to Landfill = S[5]
Distance from the materials handling facility to Market for
Other Material = S[6]
Cost Per Mile Per shipment for Glass = S[7]
Cost Per Mile Per Shipment for Newspaper = S[8]
Cost Per Mile Per Shipment for Aluminum = S[9]
Cost Per Mile Per Shipment for Plastic =
S[10]

Cost Per Mile Per Shipment for Solid Waste = S[11]
Cost Per Mile Per Shipment for Other Material = S[12]
Weight Capacity Per Shipment of Glass = S[13]
Weight Capacity Per Shipment of Newspaper = S[14]
Weight Capacity Per Shipment of Aluminum = S[15]
Weight Capacity Per Shipment of Plastic = S[16]
Weight Capacity Per Shipment of Solid Waste = S[17]
Weight Capacity Per shipment of Other Material = S[18]

Tipping Fees = DIS[1]
Government Subsidy = DIS[2]

Processing Equipment Cost = PP[1]
Direct Process Costs = PP[2]
Processing Facility Monthly Overhead = PP[3]
Number Operators at Processing Plant = PP[4]
Process Efficiency Ratio = PP[5]

Price Per lb of PE = PR[1]
Price Per lb of Clear PET = PR[2]
Price Per lb of Colored PET = PR[3]
Price Per lb of Clear Glass = PR[4]
Price Per lb of Green Glass = PR[5]
Price Per lb of Brown Glass = PR[6]
Price Per lb of Aluminum = PR[7]
Price Per lb of Newspaper = PR[8]
Price Per lb of Mixed PE/PET = PR[9]
Price Per lb of Mixed Glass = PR[10]
Price Per lb of Other Material = PR[11]
Selling Mixed Plastic = PR[12]
Selling Mixed Glass = PR[13]

Payout/month = CP[1]
Collection Point Equip Cost/month = CP[2]
Overhead /month = CP[3]
Total Collection Point Cost/month = CP[4]

Total Solid Waste/month = RM[1]
Recyclable Aluminum/month = RM[2]
Recyclable Newspaper/month = RM[3]
Recyclable Glass/month = RM[4]
Recyclable Plastics/month = RM[5]
Recyclable Other Material/month = RM[6]
Recyclable PE/month = RM[7]
Recyclable PET/month = RM[8]
Recyclable Clear PET/month = RM[9]

```
Recyclable colored PET/month =       RM[10]
Recyclable Clear Glass/month =       RM[11]
Recyclable Green Glass/month =       RM[12]
Recyclable Brown Glass/month =       RM[13]
Solid Waste Displacement[wt]=        RM[14]
Recyclable Solid Waste/month =       RM[15]
Post Recycled Solid Waste/month =    RM[16]

Avg Miles/Collection Point =         CS[1]
Labor Cost/month =                   CS[2]
Equip Cost/month =                   CS[3]
Overhead Cost/month =                CS[4]
Mileage Cost/month =                 CS[5]
Total Cost of Collection =           CS[6]

Total Number of Operators =            MH[1]
Materials Handling Labor Cost/month = MH[2]
Material Handling Equip Cost/month =  MH[3]
Total Materials Handling Cost =       MH[4]

Shipping Cost for Aluminum/month =       SH[1]
Shipping Cost for Glass/month =          SH[2]
Shipping Cost for Newspaper/month =      SH[3]
Shipping Cost for Plastics/month =       SH[4]
Shipping Cost for Other Material/month= SH[5]
Shipping Cost for Solid Waste =          SH[6]
Total Recyclable Shipping Cost/month =  SH[7]

Total Tipping Fees =         DI[1]
Tipping Fee Credit[wt] =     DI[2]
Gov Credit =                 DI[3]
Tipping Fee Credit[vol] =    DI[4]

Plastics Processing Labor Cost/month =  PL[1]
Total Plastics Processing Cost/month =  PL[2]

PE Revenue/month =                   RE[1]
Clear PET Revenue/month =            RE[2]
Colored PET Revenue/month =          RE[3]
Clear Glass Revenue/month =          RE[4]
Brown Glass Revenue/month =          RE[5]
Green Glass Revenue/month =          RE[6]
Aluminum Revenue/month =             RE[7]
Newspaper Revenue/month =            RE[8]
Mixed PE/PET Revenue/month =         RE[9]
Mixed Glass Revenue/month =          RE[10]
Other material Revenue/month =       RE[11]
Total Revenue =                      RE[12]
Post Recycled Solid Waste/month =    RE[13]
Tot Glass Rev =                      RE[14]
Tot Plastic Rev =                    RE[15]

Recyclable Collection Cost/month =     CR[1]
NonRecyclable Collection Cost/month = CR[2]
```

Recyclable Material Handling Cost = CR[3]
NonRecycling Material Handling Cost = CR[4]
Total Recycling Revenue = CR[5]
Total Recycling Cost = CR[6]
Total Non Recycling Cost = CR[7]
Net Revenue = CR[8]
Adjusted Total = CR[9]

Recyclable Other Material Vol = NC[1]
Recyclable Aluminum Vol = NC[2]
Recyclable Glass Vol = NC[3]
Recyclable Plastics Vol = NC[4]
Recyclablel Newspaper Vol = NC[5]
Recyclable Vol = NC[6]
Total Solid Waste Vol = NC[7]
Collect Pt Plastic = NC[8]
Collect Pt Glass = NC[9]
Collect Pt Aluminum = NC[10]
Collect Pt Newspaper = NC[11]
Collect Pt Other Material = NC[12]
Plastic Collection System Cost = NC[13]
Glass Collection System Cost= NC[14]
Aluminum Collection System Cost = NC[15]
Newspaper Collection System Cost = NC[16]
Other Material Collection System Cost = NC[17]
Plastics Material Handling Cost = NC[18]
Glass Material Handling Cost = NC[19]
Aluminum Material Handling Cost = NC[20]
Newspaper Material Handling Cost = NC[21]
Other Material Handling Cost = NC[22]
Total Plastic Cost = NC[23]
Total Glass Cost = NC[24]
Total Aluminum Cost = NC[25]
Total Newspaper Cost = NC[26]
Total Other Material Cost = NC[27]
Plastic Recycling Cost Per lb = NC[28]