



This is to certify that the
thesis entitled
THE COMPUTER PROGRAM DEVELOPMENT
FOR THE SWIPES MODEL
presented by

Natgritta Taveesook

has been accepted towards fulfillment
of the requirements for

Master degree in Packaging

Major professor

Date 5/28/93



PLACE IN RETURN BOX to remove this checkout from your record.
TO AVOID FINES return on or before date due.

DATE DUE	DATE DUE	DATE DUE
MAY 18 2002	_____	_____
DEC 18 2009	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____

MSU is An Affirmative Action/Equal Opportunity Institution

c:\pic\dateduea.pm3-p.1

**THE COMPUTER PROGRAM DEVELOPMENT
FOR THE SWIPES MODEL**

By

Natgritta Taveesook

A THESIS

**Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of**

MASTER OF SCIENCE

School of Packaging

1997

ABSTRACT

THE COMPUTER PROGRAM DEVELOPMENT FOR THE SWIPES MODEL

By

Natgritta Taveesook

Nowadays, packages have necessary roles in all products we concern. Although packaging can show great functions, it releases many environmental problems in its waste stream. Solid Waste integrated Packaging Evaluation System was developed by Brian Saputo as his Master's thesis to determine the volume of packaging waste. The goal of this thesis is to develop software for SWIPES Model that can be used in the packaging business. Because this model will be used by packaging engineers and any other concerning people, it should run easily and conveniently. The Visual Basic language is considered. The software includes a data file, which will be able to update at any time, a welcome screen, and programs to evaluate the value for different materials. In this software, users not only can choose the data provided in the updated file, but also they can input their own data. The calculated results can be printed out or stored to recall later.

ACKNOWLEDGMENTS

I wrote this, but a lot of people's influences are in it. And there are others who may not have contributed to the thesis, but have contributed to me.

First of all, there are my three committees: Chairperson Dr. Susan Selke, and Dr. Diana Twede of the School of packaging, and Dr. Matt Mutka of the College of Engineering without whom this would not be possible.

My parents and my family have supported me to study in Michigan State University. Their importance in both emotional way and financial way help me reach my achievement.

Then I would like to thank you all my friends: Mr. Natawut Nupairoj who helped me about computer problems, Mr. Chatpetch Saipetch who corrected my English, Chatchai who tried to understand me and all others who helped me through my obstructions.

Table of Contents

	Page
Chapter 1: Introduction	1
Chapter 2: Computer Programming and Visual Basic Language	17
Chapter 3: Computer Program for SWIPES Formula	28
Chapter 4: Conclusions	42
List of References	44
Additional Sources for Computer Programming	45
Appendix A: Computer Source Code for SWIPES	46
Appendix B: Material Data in Standard Data File	137

List of Figures

	Page
Figure 1.1: US population and municipal solid waste generation, 1960-1993	4
Figure 1.2: Products generated in MSW by weight, 1993 (Total Weight 206.9 million tons)	5
Figure 1.3: Material generated in MSW by weight, 1993 (Total Weight 206.9 million tons)	6
Figure 1.4: Landfill volume of products in MSW, 1993 (in percent of total)	7
Figure 1.5: Landfill volume of materials in MSW, 1993 (in percent of total)	8
Figure 2.1: Machine language	18
Figure 2.2: Assembly language	19
Figure 2.3: Sample of a Visual Basic program	20
Figure 2.4: Main windows of Visual Basic design screen	22
Figure 2.5: The menu window in Visual Basic design screen	22
Figure 2.6: Form window	23
Figure 2.7: Toolbox window	24
Figure 2.8: Project window	25
Figure 2.9: Properties window	26

List of Figures (Continued)

	Page
Figure 2.10: Code window	27
Figure 3.1: Main menu screen	29
Figure 3.2: About SWIPES program screen	29
Figure 3.3: Product and packaging information screen	31
Figure 3.4: Component information screen	32
Figure 3.5: Material menu screen	33
Figure 3.6: Material information screen	34
Figure 3.7: Packaging production information screen	35
Figure 3.8: Arbitrary value information screen	36
Figure 3.9: Material SWIPES value screen	37
Figure 3.10: Total SWIPES value screen	38
Figure 3.11: SWIPES value file selection screen	39
Figure 3.12: Data file selection screen	41
Figure 3.13: Data information screen	41

CHAPTER I

INTRODUCTION

1.1 Packaging

Packaging is always differently defined by many people depending on how they deal with it. From those definitions, one of the best is given in the "Glossary of Packaging Terms" (Manypenny et al, 1996) as following;

"Packaging is the enclosure of products, items or packages in a wrap, pouch, bag, box, cup, tray, can, tube, bottle, or other container form to perform one or more of the following major functions.

1. Containment for handling, transportation and use.
2. Preservation and protection of the contents for required shelf and use life and sometimes protection of the external environment from any hazards of contact with the contents.
3. Identification of contents, quantity, quality, and manufacturer-usually by means of printing, decoration, labelling, package shape or transparency. Usually the design and decoration must facilitate selection and motivate sales, and increasingly they must meet government requirements including ingredients listing, nutritive value and the like.
4. Facilitate dispensing and use, including case or opening, reclosure (if required), portioning application, unit-of-use, multipacks, safety, second use or re-use and working features such as are found in aerosol sprays, cook-in-bag, "memory packs" and especially provision for instructions or directions.

If the device or container performs one or more of these functions it is considered a package. Packaging is also the development and production of packages (filling, closing, labelling) by trained professionals or operators employing methods and equipment designed for specific product lines and types of packages. Some 30 different categories of packaging machines are employed, for example, fillers, counters, cappers, labelers, wrapping equipment, cartoners, case loaders, and sealers, as well as a wide range of support equipment for

package making, inspection, monitoring, handling, and the like. Increasingly, packaging machinery is designed and integrated to provide a complete system.

Packaging is often referred to as consumer type, industrial type, military type, shipping type, and the like, designations that are more or less self-explanatory. Consumer packages are intended for retail merchandising and emphasize sales features. Industrial packages are intended for use in manufacturing, services and repair outlets. Sometimes characteristics and functions are overlapping and terminology is used loosely."

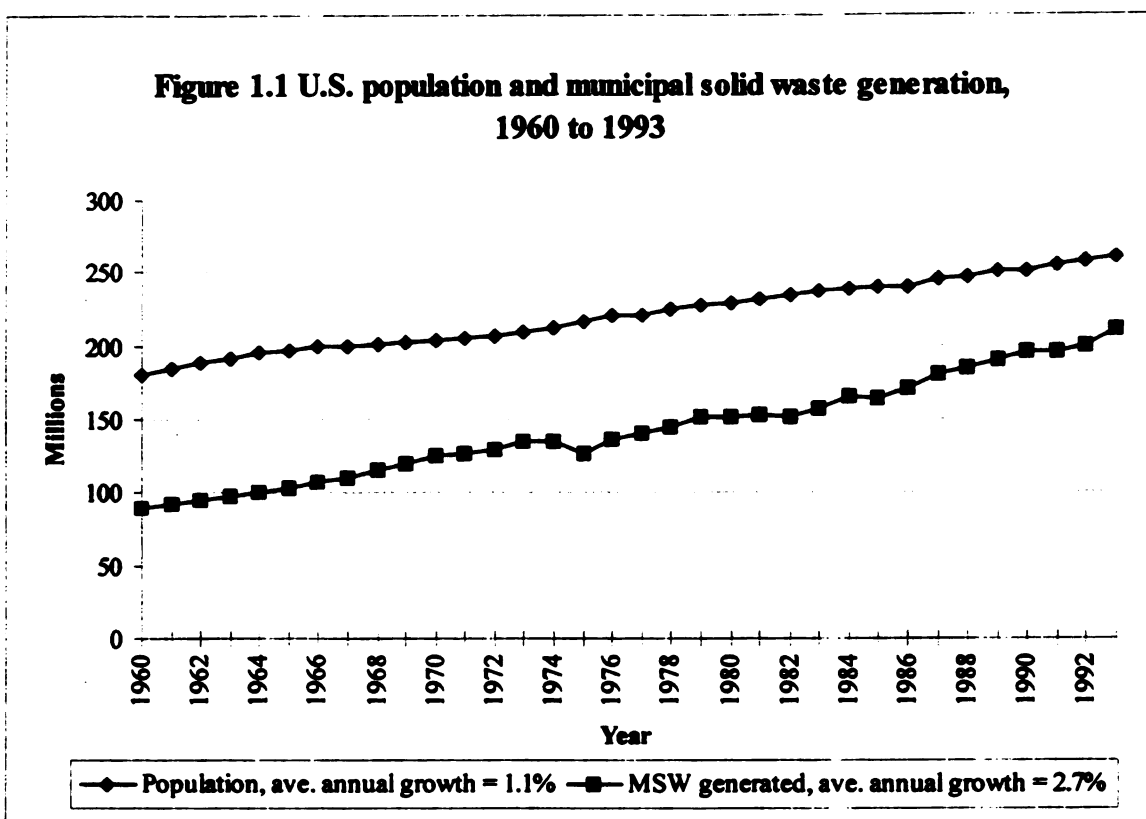
Packaging (Abbott, 1989) has been a part of the human race since the beginning of its existence. Although quite simple in technology at first, the development of the process itself is as extraordinary as that of any other technology. The primitive packages were simple in design and were created for transporting easily gathered and handled foods. The evolution of primitive people into highly cultured and industrialized societies created a demand for more advanced and specialized packaging that is the norm today. The requirements placed on modern packaging containers not only include that the contents be held together physically, but also may involve the preservation of other chemically sensitive qualities which can be infinitely more inconstant than physical appearance. And as our population grows, the need for new and more efficient forms of packaging will increase accordingly.

Today, treated papers, metals and synthetic materials make up the core of the available packaging systems. Instead of containers created from mollusk shells and bamboo, modern packaging science now employs a variety of materials from timber and steel to plastics and revolutionary composite materials. The pace of change within the world of packaging (Abbott, 1989) really intensified after 5 revolutionary advances (metal cans, collapsible tubes, folding cartons, corrugated shipping cases, and crown closures) were introduced (during the Industrial Revolution). However, the introduction

of these modern materials does have its disadvantages. The increasing rate of municipal solid waste is one of the most important problems facing us today.

1.2 Solid Waste Problems From Packaging

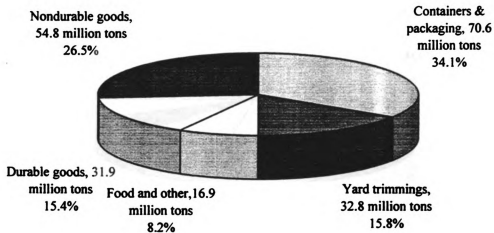
Municipal solid waste (MSW), as defined by the United States Environmental Protection Agency (EPA), includes wastes such as durable goods, nondurable goods, containers and packaging, yard trimmings, and food waste. To eliminate the MSW, many processes including reuse, recycling, combustion, and incineration are used together in one combination or another. However, all of these processes are still not capable of absolutely eliminating MSW. The remaining waste is sent to the process known as landfill, a method of solid waste disposal in which refuse is buried between layers of dirt. EPA (1994) records show that from 1960 to 1993, while the population of the United States was annually increasing at the rate of 1.1 %, the annual growth of MSW was 2.7 percent (from 88 million tons per capita to 207 million tons per year), illustrated in Figure 1.1. This trend cannot be allowed to continue. Our current level of technology does not permit us the luxury of producing that great amount of waste and still expect the same proportion of generation for all eternity. A way of dealing with the problem of MSW is crucial to the livelihood of future generations. Thus, is the birth of the field of Solid Waste Management.



Source: EPA *Characterization of Municipal Solid Waste in the United States: 1994 Update*,

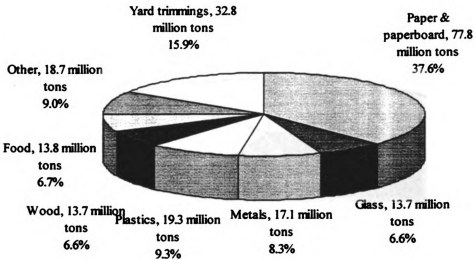
MSW is divided into 3 main categories: durable goods, non-durable goods, and containers and packaging. In 1993 (EPA, 1994), containers and packaging equalled 70.6 million tons or 34.1% of the wastes generated (Figure 1.2). Paper and paperboard, some of which are included in the containers and packaging category (EPA, 1994), accounted for 38% (by weight), Figure 1.3. Other packaging materials contributing to the total MSW included glass, metals, plastics, and wood, and ranged from 6 to 9 percent (by weight.)

Figure 1.2 Products generated in MSW by weight, 1993 (Total weight = 206.9 million tons)



Source: EPA *Characterization of Municipal Solid Waste in the United States: 1994 Update*

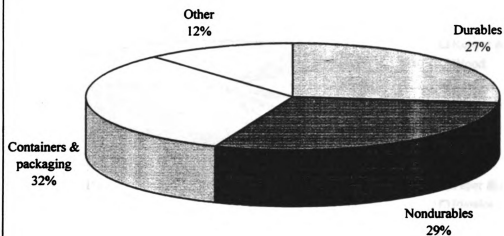
Figure 1.3 Materials generated in MSW by weight, 1993 (Total weight = 206.9 million tons)



Source: EPA *Characterization of Municipal Solid Waste in the United States: 1994 Update*

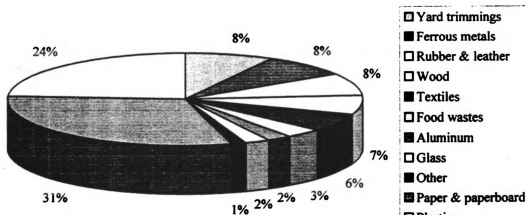
The waste produced from these packaging materials is processed through a system of waste management programs; however, even with all of today's technology, packaging remained the largest contributor after recycling and composting, Figure 1.4 (EPA, 1994). Again, in Figure 1.5, paper & paperboard and plastics were the top of the list at 30.2% and 23.9% respectively (EPA, 1994).

**Figure 1.4 Landfill volume of products in MSW, 1993
(in percent of total)**



Source: EPA *Characterization of Municipal Solid Waste in the United States: 1994 Update*

**Figure 1.5 Landfill volume of materials in MSW, 1993
(in percent of total)**



Source: EPA *Characterization of Municipal Solid Waste in the United States: 1994 Update*

EPA (1994) states that integrated solid waste management emphasizes source reduction of wastes before they enter the waste stream, recovery of generated wastes for recycling and composting, and environmentally sound disposal through combustion facilities and landfills that meet specified standards. The four stages are defined as follow (Saputo, 1991.)

“Source Reduction: The optimum way to reduce waste is by not producing it in the first place. Source reduction includes: reducing the amount of packaging materials used, developing products with longer useful lives, utilizing reusable and multi-use packages, and selecting the appropriate materials for the application.

Recycling: Recycling involves the separation and marketing of a waste material (e.g. paper, glass, aluminum) from the waste stream so that it can be remanufactured into a usable product.

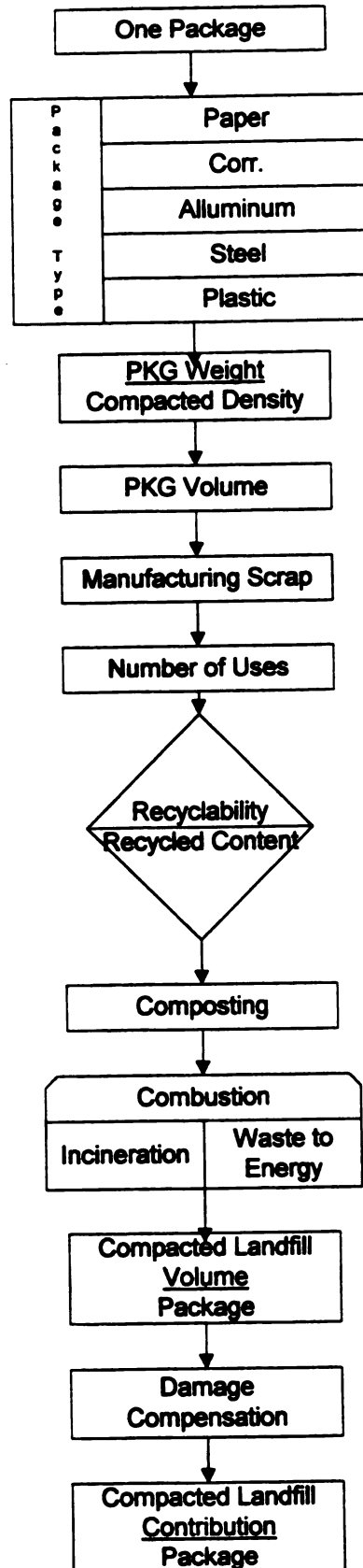
Combustion: Incinerators can burn wastes strictly to reduce the volume of waste that must be landfilled. Waste-to-energy incinerators also reduce the volume of waste, but these facilities are designed to recover energy in the form of steam or electricity.

Landfilling: This option involves burying waste in “sanitary” landfills. It should only be necessary (in an optimum system) for items that cannot be managed by the other three approaches.”

1.3 SWIPES Model

SWIPES stands for Solid Waste Integrated Packaging Evaluation System. This model was derived and written by Brian Saputo as his thesis project for the Master of Science degree, submitted to the School of Packaging, Michigan State University. Saputo said in his thesis that “The goal of the SWIPES model is to quantify the volume of waste that the package can contribute to a landfill.” In this model, volume, compaction density, number of uses, recyclability, combustion, composting, manufacturing scrap, and protective effectiveness were considered and used in quantitative terms (because landfill is affected more by volume than by weight.) This model works based on the following flowchart:

SWIPES Flowchart



Variables affecting the SWIPES formula can be interpreted in many different ways. For the consistent understanding of each variable's effect on the model, all will be briefly discussed.

SWIPES Equation

$$\underline{s} = \frac{(m_{pkg}/d_{pkg})[1+f(m_s/m_{pkg})](1-X_1r_r)(1-X_2r_c)[1-(Yc)(1-e)][1-(Zi)(1-a)][1+b] + P + S}{1 + W(n-1)}$$

Note: $W \geq X \geq Z$ and $W \geq Y \geq Z$ are suggested.

where:

a = fraction of ash after combustion

b = average percent product damage in distribution

c = composting rate

d_{pkg} = compaction density of package

d_{pr} = compaction density of product

e = residue after compost screen

f = fraction of scrap disposed of

i = incineration rate

m_{pkg} = mass or weight of package

m_{pr} = mass or weight of product

m_s = mass or weight of scrap per package

n = number of uses

P = product correction factor

= b(SWIPES value for product) or

= $b(m_{pr}/d_{pr})[1 - (iZ)(\% \text{ combustible of product})]$

r_c = recycled content

r_r = recycling rate

S = Correction factor for manufacturing scrap

= $f(m_s/d_{pkg})(X_1)(r_r)$

\underline{s} = SWIPES

W = arbitrary value for reuse ($0 \leq W \leq 1$)

X_1 & X_2 = arbitrary value for recycling ($0 \leq X_1 \leq 1$) and ($0 \leq X_2 \leq 1$)

Y = arbitrary value for composting ($0 \leq Y \leq 1$)

Z = arbitrary value for combustion ($0 \leq Z \leq 1$)

Packaging Volume (m_{pkg}/d_{pkg}); The SWIPES model was designed to be based on volume because landfills are always filled up by volume, not weight. However, it is much easier to measure waste by weight as it is done in most experiments and research; therefore it became necessary to develop a method of conversion in which volume can be calculated consistently without burdensome calculations. Volume of packaging wastes is different for each form of package (1 pound of flat sheet paper and 1 pound of paper carton, for example) and also differs according to the landfill environmental conditions. For consistency, MSW volume is calculated by dividing weight by compaction density.

Compaction Density (d_{pkg}); In each landfill, waste will be compacted by heavy equipment. In general, only data on mixed wastes is available, but a study by the

EPA-Franklin Associates, 1990 update, found that it is possible to get some information of the ideal separated waste stream from the type of waste involved.

Reuse $[1 + W(n-1)]$; The compacted volume is divided by this factor to credit the package if it has multiple use cycles. W is an arbitrary factor which represents discounting of a portion of the reuse credited to account for the environmental impact of reuse. Some types of packaging that have been found successful as reusable packages are glass bottles, industrial returnable packaging, and distribution packaging. It is possible for a packaging engineer to design a package that can be used more than once. However, the problem of reusable packages is returning the packages to a goods manufacturer who can reuse them. Cleaning and reconditioning may also be required. Although the industrial returnable packaging and distribution packaging systems are growing rapidly, a more efficient return system must be developed to increase the effectiveness of this method as a part of the source reduction of packaging solid waste. It is also vital that similar processes be developed for other packaging material.

Recycled Content $(1-X_2r_c)$; The compacted volume is multiplied by this factor to credit the packaging component if it has a portion of recycled material. X_2 , like W , is an arbitrary value which represents discounting of a portion of recycled content contained in the material. To reduce packaging materials going to the waste stream, every attempt should be made to incorporate recycled materials into the production of new packaging. The emphasis should be to maintain the highest possible level of secondary materials (derived from post-consumption residuals) within the production. Most packages, except glass and metal, cannot be produced at 100% recycled content because those materials always have different physical and chemical properties from a

virgin one. Moreover, the problems of using recycled materials are their costly processing and the migration of contaminants from recycled contents, which might cause an undesirable change in products, especially in foods and drugs.

Recycling Rate $(1-X_1r_r)$; The compacted volume is multiplied by this factor to credit the packaging component if it can be recycled after use. X_1 is an arbitrary value which represents discounting of a component with recycling capability. By theory almost all packaging materials can be recycled, but practically, only some materials are put through a recycling process efficiently while others might be never recycled at all. The recycling rate of each material is controlled by its feasibility of cost and technology in processes, and its use after recycling. Some packaging materials, especially composite materials, though they can be technically recycled, need to be broken up first, which is impractical, and some even if it is practical to recycle, are not stable enough to be processed in the next steps. Therefore, even through the recycling rate of many materials is measurable, at present, those recycled materials are not thoroughly beneficial until manufacturers have enough capability to use them.

Combustion $[1-(Z_i)(1-a)]$; The compacted volume is multiplied by this factor to credit the packaging component if it can be incinerated after use. Z is an arbitrary value which represents discounting of waste reduction by combustion. This has the goal of reduction of waste volume by burning the organic flammable part of the waste. There are two burning systems used at present: incineration which is the old style, tends to be dirty, and operates with little or no pollution control; and waste-to-energy facilities which are not only designed for volume reduction but also focus on energy recovery from burning waste.

Composting $[1-(Y_c)(1-e)]$; The compacted volume is multiplied by this factor to credit the packaging component if it can be composed after use. Y is an arbitrary value which represents discounting of a component with composting capability (with some organic substances). In this step, a large portion of organic waste can be separated from the waste stream and be decomposed. Packaging materials that are involved in composting most are papers. Anyway, as mentioned before, paper and paperboard are the biggest groups in the waste stream; thus the reduction of paper volume discarded will have a great effect on MSW amount.

Manufacturing Scrap $[S, 1+f(m_s/m_{pkg})]$; This factor is included in the formula to account for scrap produced during the packaging production. In any production line, the efficiency of every machine and efficiency of using material supplies can never reach 100 percent. Then some of the remaining supplies might be recycled and the others might be dumped as scrap. The scrap which is disposed of from those processes is defined as manufacturing scrap.

Damage Compensation $(1 + b)$; Like the manufacturing scrap factor, this factor is a multiplier of the compacted density to account for the value of the packages which are excessively produced because of damages during the distribution. In many cases, packaging is designed to use more materials than is necessary for containment alone, to make sure that the package is strong enough to protect goods until customers get and use them. Besides the quality of the products, the protection properties of packages are very important to the amount of the waste stream. If the product fails because of inadequate protection by packaging, both product and package must go to the waste stream. That means the MSW is unnecessarily increased. However, the amount of

materials used in a packaging system tends to be minimized because of the development of new technology and new synthetic materials which give lighter and stronger packages without compromising protection.

Product Correction Factor $[P, b(m_{pr}/d_{pr})[1-(iZ)(\% \text{combustible of product})]]$;

This value can be defined as the SWIPES value of the product included in packaging systems. Therefore, this factor is added once into the whole packaging system. However, some of the product variables, such as reuse and recycling rate, are not available or very small; thus the factor cannot be calculated by the same formula as the packaging materials. Therefore, it is simplified by involving its compaction density and combustion alone.

1.4 The Need for the Computer Program

As shown above, the SWIPES equation is composed of many comprehensive parts. There are almost 20 factors we need in the one equation. Even for only one packaging component, it really takes time to calculate this value. Unfortunately, most packaging systems in the world would never be a single component. Therefore, a computer program will minimize the time we lose in packaging design. However, as Saputo said, the studies about solid waste management are still not completed. More study might help us get better solutions and more comprehensive equations. This computer program should be useful during the period of study. Then after we have a developed model, because this program is written by a user-friendly computer language, it can be easily revised.

CHAPTER II

COMPUTER PROGRAMMING AND VISUAL BASIC LANGUAGE

2.1 Introduction to Visual Basic

Visual Basic (abbreviated “VB”, (Millard, 1997)) is a programming language initially developed by the Microsoft Company. This language has an ability to provide programmers with a quick and easy method of developing Windows applications. Visual Basic provides the programmers with an integrated environment where they can use tools to create a point and shoot interface and use event-driven programming techniques. A programmer can quickly and easily create a user interface, then write the code to respond to specific events which occur as a result of user input. Currently, there are two different “flavors” of Visual Basic; the original which was developed for Windows, and Visual Basic for DOS. Visual Basic for Windows gives the programmers the capability of creating applications which run in Windows, while Visual Basic for DOS gives the ability to develop an application which does not need Windows to runs.

Visual Basic (VB) is one of the new computer languages developed in the world. Just as human beings communicate with each other through the use of languages like English, French, Spanish, or Thai, programmers communicate with computers through

special programming languages which are comparably varietal. Each programming language differs from others and has its own structure and grammar. For example, VB is more visually oriented than the others, and might be thought of as a sort of sign language (Zak, 1995). However, most computer programming languages basically have the same development history.

The development of programming technology resulted in many levels of computer languages :

A **Machine Language** (Zak, 1995) is the very first computer programming method in the world, and uses on and off electrical systems to create a series of machine instructions.

Figure 2.1 Machine language

```
0100
001101 100000 001101 110001
00101 10001 10000
01110
111001
111001 001 11000 001
11000
0011100
100010 00110
```

Source: Zak, Diane *Programming with Microsoft Visual Basic 2.0/3.0 for Windows*, 1995

An Assembly Language (Zak, 1995) uses mnemonics (alphabetic abbreviations) instead of the on-and-off system. It needs a program called the assembler to convert the assembly instructions into machine code.

Figure 2.2 Assembly language

```
main proc pay
    mov ax, dseg
    mov ax, 0b00h
    add ax, dx
    mov al,bl
    mul bl,ax
    mov bl,04h
```

Source: Zak, Diane *Programming with Microsoft Visual Basic 2.0/3.0 for Windows*, 1995

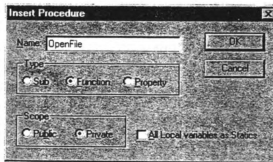
A High-Level Language's instructions (Zak, 1995) are closer to the English language. They require either an interpreter (translating the high-level code into machine code line by line while running the program) or a compiler (translating the entire code into machine code before running the program.) Most high-level languages are procedure-oriented (the sequence of instructions is followed step by step.)

An Object-Oriented/Event-Driven High-Level Language (Perry, 1995) simplifies the task of programming application for Windows by providing a graphical user interface (GUI). This system is very user-friendly because it allows for interaction between the user and whatever they see, but it requires more difficult and complicated programming tools. The emphasis of a program is on the objects in an interface (such as

a command button and a scroll bar) and the events occurring on those objects (such as clicking and scrolling).

Visual Basic (VB) can be classified as an Object-Oriented/Event-Driven Language (Perry, 1995). One of the most important components of VB is its procedural Basic-like (Beginner's All-Purpose Symbolic Instruction Code) programming language underneath the visual environment. Each part of the program is broken up and focus is placed on each object independent from others. In addition, all executable codes in a VB program exist in either a subprogram or function that is activated by events or by calling from other routines. Figure 2.3 is a sample of a Visual Basic program.

Figure 2.3 Sample of a Visual Basic program



```
Private Function OpenFile(fileName As String) As Boolean
Dim FNum As Integer
On Error GoTo OpenFileError
FNum = FreeFile
Open fileName For Input As FNum
txtFile.Text = Input(LOF(FNum), #FNum)
Close FNum
m_fFileDirty = False
m_strFileName = fileName
OpenFile = True
Exit Function
OpenFileError:
MsgBox Err.Description, vbExclamation
OpenFile = False
Close
End Function
```

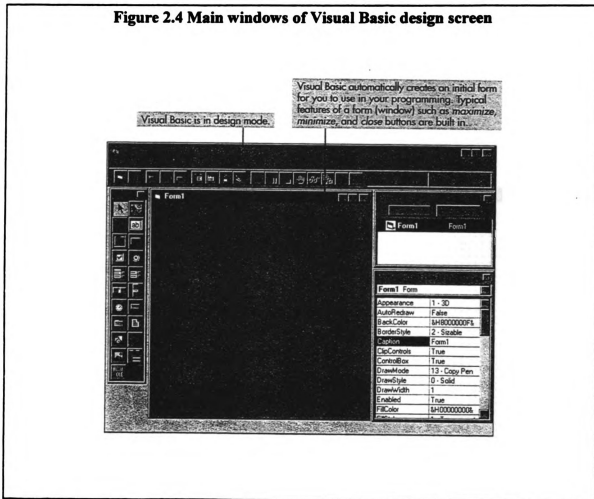
2.2 Strengths and Weaknesses of Visual Basic

Visual Basic's great strength lies in its ability to enable a programmer to quickly and effortlessly develop application interfaces. The interface being developed is exactly what users will see. Visual Basic's Rapid Application Development (RAD) environment (Norton, 1996) is useful for rapidly refining interface design and development approaches. Additionally, in today's world, applications must be written in such a way that they can be easily changed in order to keep them from being obsolete. For example, a tax-processing application needs to be changed very often to accommodate the provisions of the Internal Revenue code. Visual Basic is a perfect tool for developing this kind of application.

However, there are disadvantages inherent to Visual Basic. The most obvious one is that Visual Basic is an interpreted programming language, which means the code is interpreted on-the-fly. Although all Visual Basic programs are first compiled to pseudo-code, called p-code, this p-code must be interpreted during the run-time by an external library called VBRUNx00.DLL (VBRUN300.DLL in this thesis.) Thus, a VB application cannot be run outside Windows and its speed is much slower than applications developed from other programming languages. However, as Windows becomes standard on most IBM/PC compatible systems and computers' speeds have often been drastically increased, these drawbacks become insignificant.

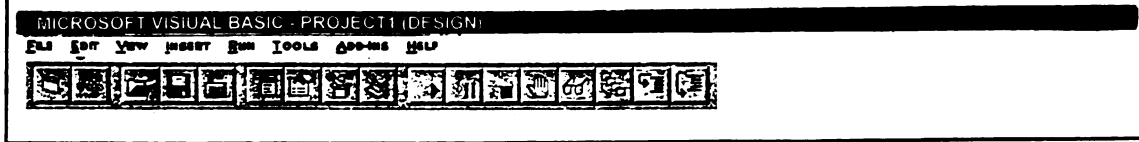
2.3 Visual Basic Environment and Programming

In VB, programmers work with several open windows most of the time (Perry, 1995). Information from different opened windows always needs to be used together. Figure 2.4 illustrates the major part of the Visual Basic screen and each of the 5 primary windows is described as follows:



Source: Visual Basic 3.0, Microsoft Corp., 1995

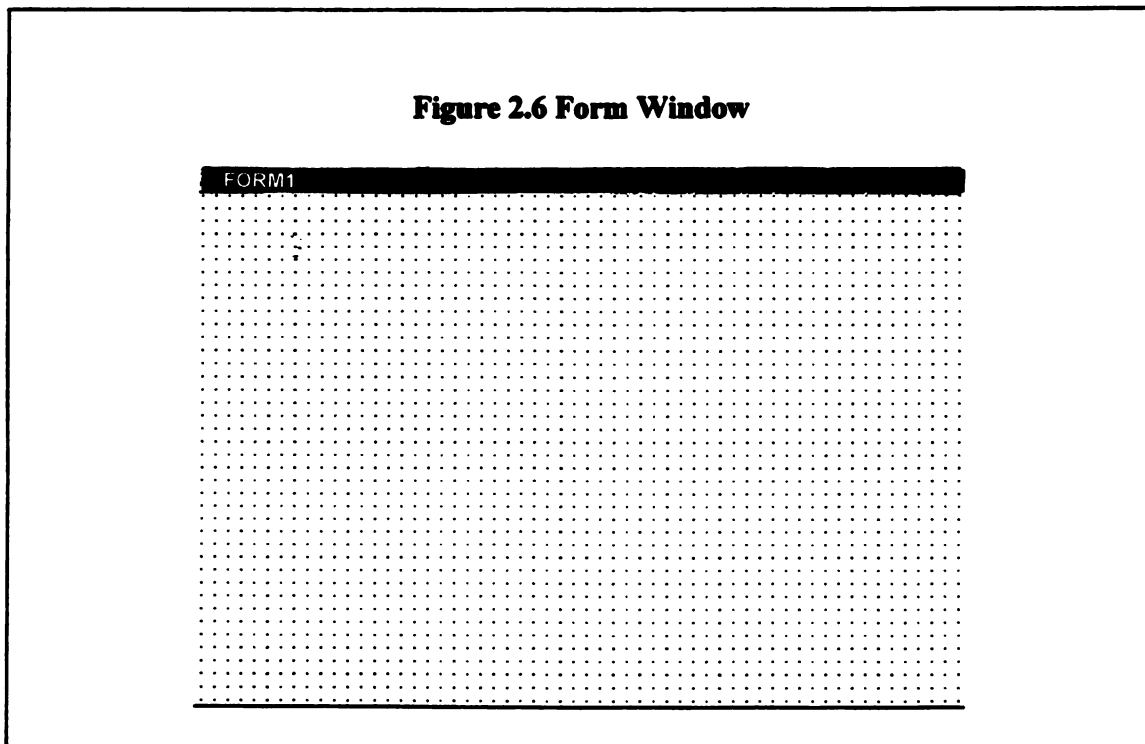
Figure 2.5 The Menu Window in Visual Basic design screen



Source: Visual Basic 3.0, Microsoft Corp., 1995

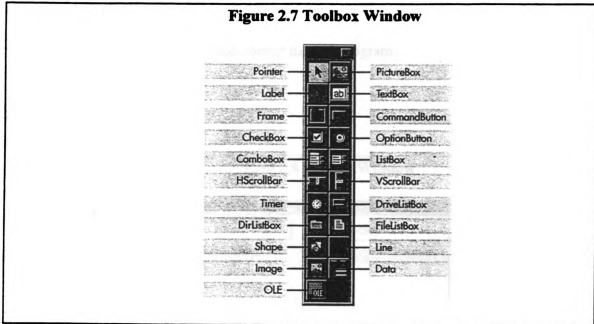
The **Form Window** (Perry, 1995) contains the background for the applications being developed. Programmers can draw and place items on the form which will be the user interface.

Figure 2.6 Form Window



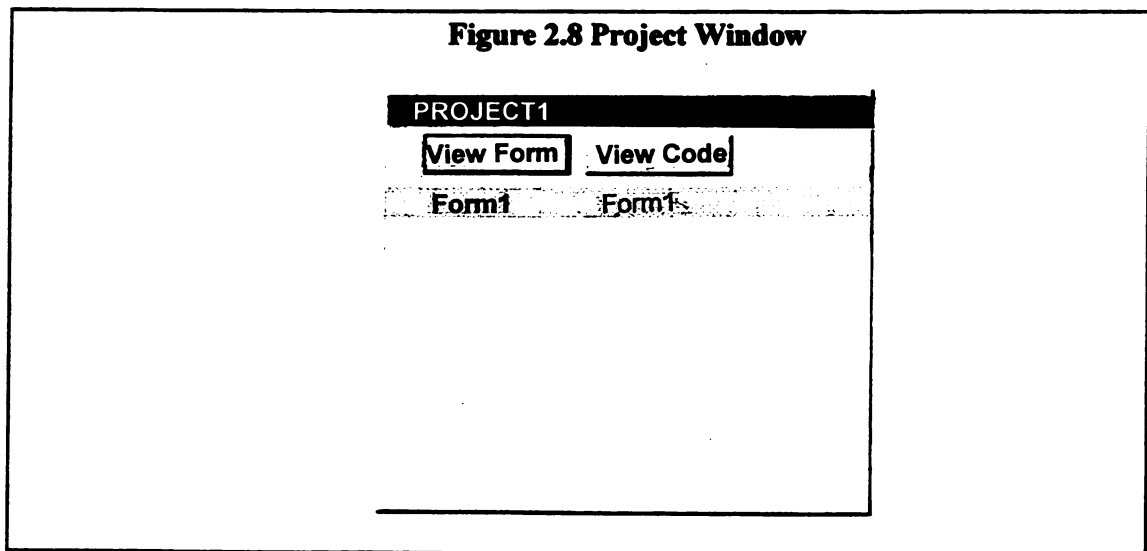
Source: Visual Basic 3.0, Microsoft Corp., 1995

The Toolbox Window (Perry, 1995) contains the set of tools (or controls) used to design a Visual Basic application. These tools allow programmers to place an object on the form.



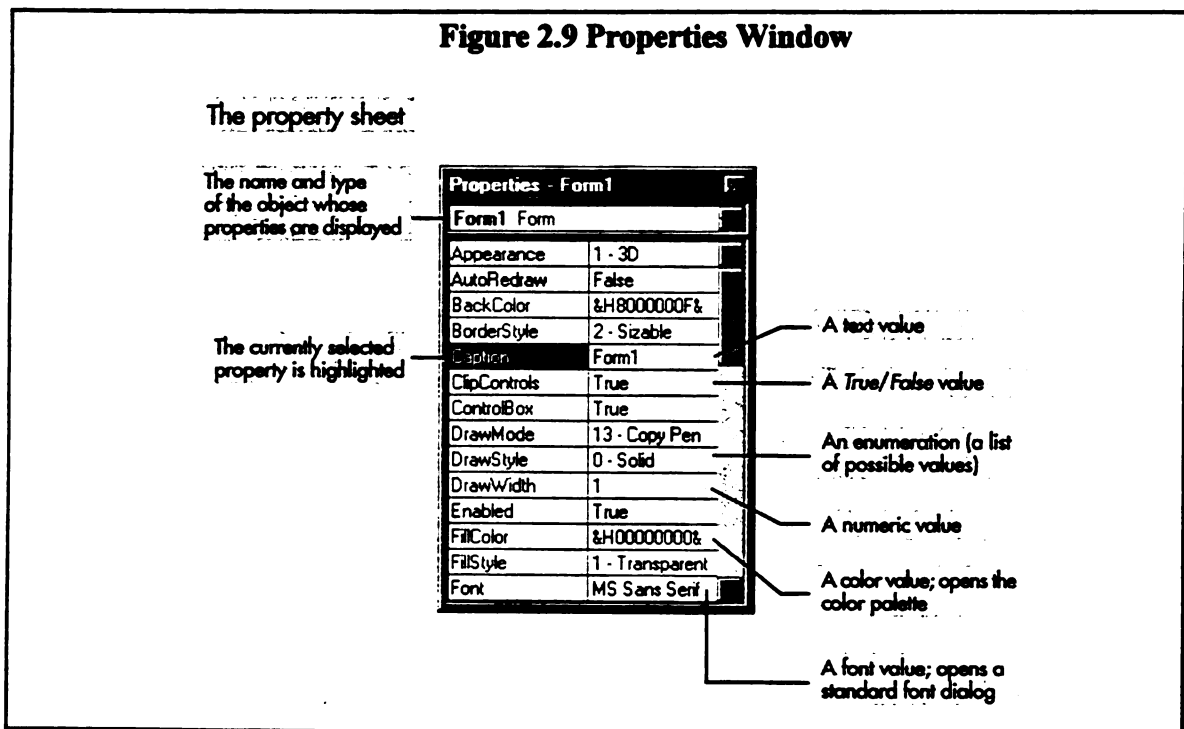
Source: Visual Basic 3.0, Microsoft Corp., 1995

The Project Window (Perry, 1995) contains the list of all files used in the current application. It is easy for a small program to include all instructions together into only one file. In contrast, in a long and more complex program, having all instructions in only one file might cause complications and confusion. It is much easier to break a long program into many modules contained in different files. This programming style allows the programmer to easily check and correct his/her program.



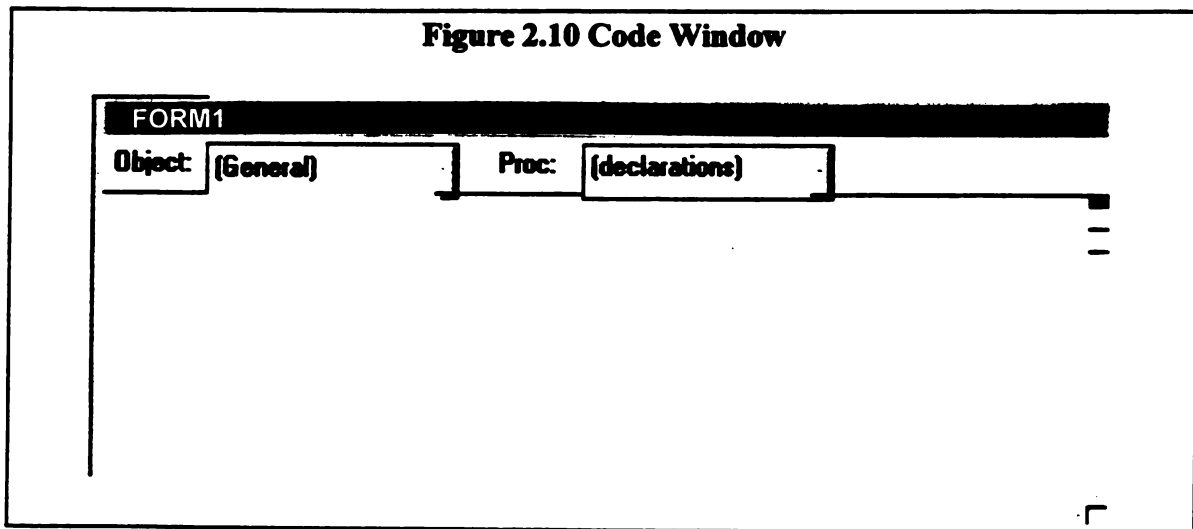
Source: Visual Basic 3.0, Microsoft Corp., 1995

The Properties Window (Perry, 1995) lists a set of characteristics associated with each object in an application, such as size and color. A programmer can control an object's appearances and activities (for example, the caption on a button) by setting that property in the properties window.



Source: Visual Basic 3.0, Microsoft Corp., 1995

The Code Window (Perry, 1995) allows programmers to describe behaviors of each object in the form when a particular event occurs. For example, a programmer has to describe how an application should respond when a user clicks a mouse on an OK button.



Source: Visual Basic 3.0, Microsoft Corp., 1995

CHAPTER III

COMPUTER PROGRAM FOR SWIPES FORMULA

The SWIPES computer program was designed to calculate the SWIPES value of packaging materials. This program is divided into 3 major sections according to their purposes: the calculation part, the SWIPES value recording part and the data recording part. Their working systems are going to be explained together with each screen.

3.1 Calculation

First, when the program is executed, its welcome screen is shown and automatically passed to the main menu screen. On the main menu screen, Figure 3.1, users have 5 different paths to choose. One option is to bypass the program by clicking the button marked "EXIT", which will shut down the program. The button marked "ABOUT SWIPES" will lead to a screen showing background and historical information of the program, Figure 3.2. The next two buttons, "SWIPES FILE" and "DATA FILE" buttons, will be discussed later in the SWIPES value recording part and data recording part.

Figure 3.1 Main menu screen

WHAT TO DO	
If you want to estimate the SWIPES value of a product click here.	SWIPES VALUE
Click here to get all information and SWIPES value from a SWIPES value file.	SWIPES FILE
If you want to create or change data in a data file click here.	DATA FILE
Click here for more information about SWIPES program.	ABOUT SWIPES
If you want to exit this program click here.	EXIT

Figure 3.2 About SWIPES program screen

ABOUT SWIPES
This computer program was written by Natgritta Taveesook and is based on the Solid Waste Integrated Packaging Evaluation System (SWIPES) model developed by Brian Peter Saputo as his thesis project for Master of Science degree, submitted to the School of Packaging, Michigan State University.
RETURN TO WHAT TO DO MENU

To calculate the SWIPES value for a set of packaging, information can be input via keyboard by clicking the “SWIPES VALUE” button or by accessing a stored data file. In the next screen, users have to enter information about the product that this packaging system will be used for. Users may use a mouse or the “Tab” button on the keyboard to position the information they want to enter. When finished, clicking the “NEXT” button will bring up the next screen. In the case of missing units, which are necessary for calculations, the program will show a warning message and allow the users to complete the information. After the information is completed, the NEXT button will take a user to the packaging component information screen. The “CANCEL” button gives a user choices to return to the main menu, to clear all information in this screen and re-enter, or to keep all information and stay on the screen.

Figure 3.3 Product and packaging information screen

PRODUCT AND PACKAGE INFORMATION	
Please enter your product and package information	
Type of product that the package is used for	<input type="text"/>
Manufacturer	<input type="text"/>
Mass of product	<input type="text" value="0"/> <div>Unit <input type="radio"/> Lb. <input type="radio"/> g. </div>
Compacted density of product	<input type="text" value="0"/> <div>Unit <input type="radio"/> Lb./in³ <input type="radio"/> g./cm³ </div>
Combustible Fraction of Product	<input type="text" value="0"/>
<p>*NOTE: Product information will be calculated only once in SWIPES formula with first material.*</p>	
<input type="button" value="CANCEL"/>	<input type="button" value="NEXT"/>

In the component information screen, users are given the options to access material information from a data file; this will be discussed in more detail in the data recording section. Users should select the type of package from the automatically loaded choices or input a new one to help with material identification (such as in case there is more than one package with identical names but differing physical properties). Clicking the "NEXT" button will show users the material menu or give warning messages if they did not input any necessary information. The "CANCEL" button, as in the product

information screen, will give a user choices to return to the main menu, to clear all information in this screen and re-enter, or to keep all information and stay on the screen.

Figure 3.4 Component information screen

COMPONENT INFORMATION

FILE INFORMATION

Please enter the component information

If you want to get the information from a data file, click here and enter a file name for each material **DATA FILE**

Type of package Bag

Type of material

Mass of package Unit
C Lb. C g.

Number of Uses

Number of Product Units per Package

CANCEL
NEXT

If you want to change any information, click the left button on your mouse at the menu bar on the higher left corner of this page

The purpose of the program is to calculate the SWIPES value of the complete packaging system of a product. In the case that there is more than one material component of the packaging system, necessary information on each component must be entered. The program will store all material data within the corresponding type the users had entered in the previous screen. For each component, the users must provide 3 types of information: material information, packaging production information, and arbitrary

value information. Users can access the necessary areas by selecting the screen from the material menu shown after the component information screen.

Figure 3.5 Material menu screen

MATERIAL MENU	
Material Menu	
1.Material Information	1
2.Packaging Production Information	2
3.Arbitrary Value Information	3
4.Calculate the material SWIPES	4
5.Return to packaging component information	5
6.Next Component from the same SWIPES value file.	6
7.Add additional components into the packaging system.	7
8.Return to What to do menu and get information from another SWIPES value file.	8

Figure 3.6 Material information screen

MATERIAL INFORMATION	
FILE INFORMATION	
Material Information	
Compacted density of package	<input type="text"/> Unit <input type="radio"/> Lb./in ³ <input type="radio"/> g./cm ³
Fraction Recycled	<input type="text" value="0"/>
Fraction Composted	<input type="text" value="0"/>
Fraction Incinerated	<input type="text" value="0"/>
Fraction of Ash after Combustion	<input type="text" value="0"/>
Residue Fraction After Compost Screen	<input type="text" value="0"/>
<p>If this information is not satisfactory, use mouse or press tab button to highlight those values and change them. Then click OK button to go back to Material Menu.</p> <p>If you want to change any information, click the left button on your mouse at the menu bar on the higher left corner of this page</p> <p>OK</p>	

Figure 3.7 Packaging production information screen

PACKAGING PRODUCTION INFORMATION

FILE INFORMATION

Packaging Production Information

Please enter the following information or all default values are zero, 0

Mass of scrap per package **g Lb g**

Fraction of scrap disposed of

Fraction of product damage in distribution

Fraction of recycled content in package component

If all values are satisfactory, click the OK button

If you want to change any information, click the left button on your mouse at the menu bar on the higher left corner of this page

Figure 3.8 Arbitrary value information screen

ARBITRARY VALUE INFORMATION

FILE INFORMATION

Arbitrary Value Information

Reuse Value, W	<input type="text" value="0"/>
Recycling Rate Value, X1	<input type="text" value="0"/>
Recycling Content Value, X2	<input type="text" value="0"/>
Composting Value, Y	<input type="text" value="0"/>
Combustion Value, Z	<input type="text" value="0"/>

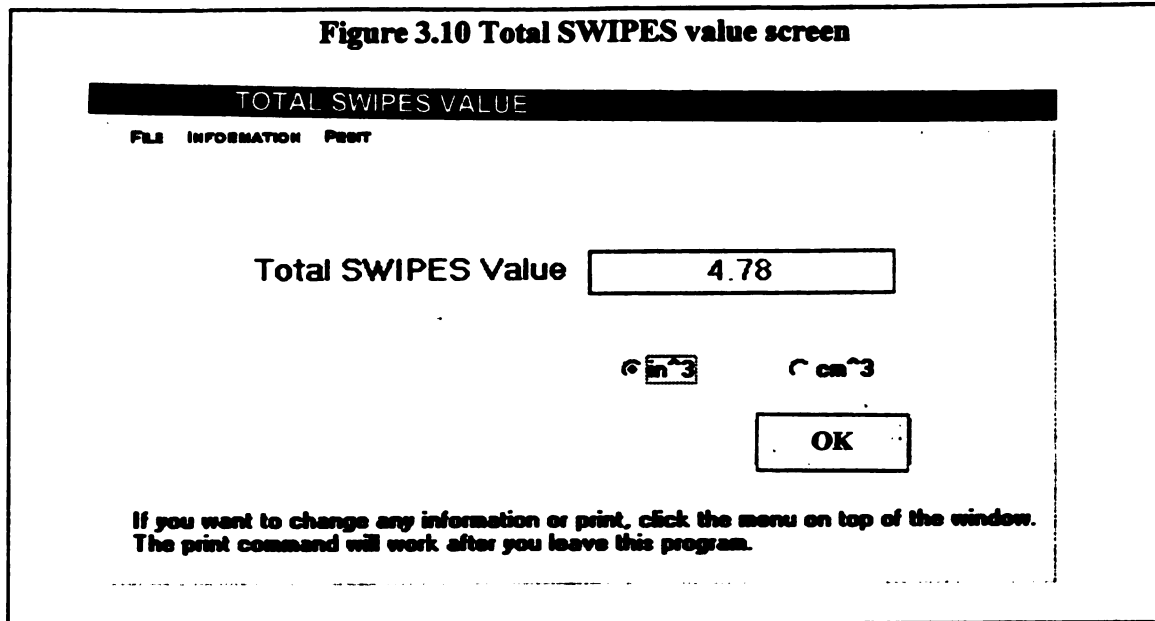
If these values are not satisfactory, use mouse or tab button to enter new value, then click OK button to return to material

If you want to change any information, click the left button on your mouse at the menu bar on the higher left corner of this page

After completing the above steps, users might choose to immediately get the results of the calculation by clicking No. 4 "Calculate the material SWIPES Value". Users will encounter the SWIPES value screen. Here, users will need to specify the materials and units before the program will perform the calculation. For a system composed of many materials, the total SWIPES value of the system will be calculated by choosing "Total SWIPES Value", which is the last topic in the list of materials. The "RETURN TO MATERIAL MENU" button will take the users back to the material menu.

Figure 3.9 Material SWIPES value screen

MATERIAL SWIPES VALUE	
FILE INFORMATION PRINT	
MATERIAL <input type="text" value="glass"/>	Show the result in <input type="radio"/> in ³ <input type="radio"/> cm ³
Compacted Volume	<input type="text"/>
Manufacturing Scrap	<input type="text"/>
Reuse	<input type="text"/>
Recycling Rate/Recycling Content	<input type="text"/>
Composting	<input type="text"/>
Combustion	<input type="text"/>
Damage Compensation	<input type="text"/>
Manufacturing Scrap Correction Factor	<input type="text"/>
Product Correction Factor*	<input type="text"/>
MATERIAL SWIPES VALUE	<input type="text"/>
<input type="button" value="RETURN TO MATERIAL MENU"/>	
<p>If you want to change any information or print, Click the left button on your mouse at the menu bar on the higher left corner of this page. The print command will work after you leave this program.</p> <p>* NOTE: Only 1 Unit of product is contained in all level of packaging. then, Product correction factor is calculated only once with the first package.</p>	

Figure 3.10 Total SWIPES value screen


TOTAL SWIPES VALUE

FILE INFORMATION PRINT

Total SWIPES Value

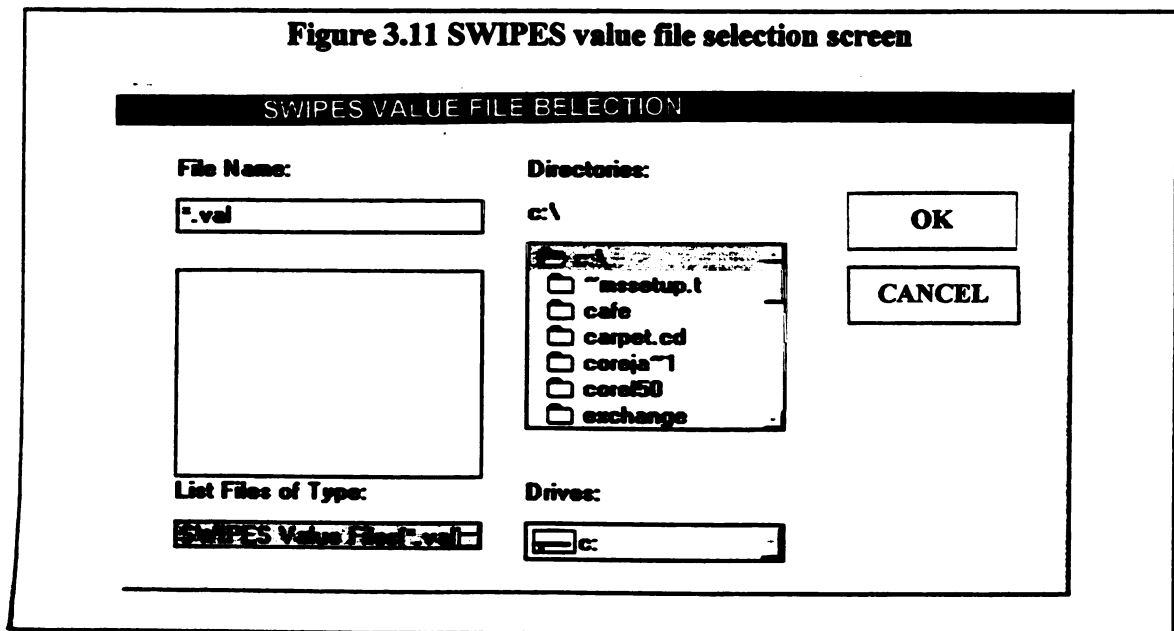
☒ in³ ☐ cm³

If you want to change any information or print, click the menu on top of the window.
The print command will work after you leave this program.

Users can also choose to go on to enter an additional component information by clicking No. 6 “Next Component from the same SWIPES value file” if users apply information and result from a SWIPES value file or No. 7 “ Add more components into the packaging system” if users want to enter additional information using a keyboard). Users can access any area of the program if they need to re-enter any data by clicking the menu bar in all information screens. Users can print the information and results of calculation by clicking “Print” in the menu bar.

3.2 SWIPES Value Recording

Users can store SWIPES values from the calculation of a packaging system by clicking “File” and entering the necessary information in the SWIPES value file selection screen. This information will be stored in the SWIPES value file. Data from the SWIPES value file can be recalled in the main menu screen by clicking the “SWIPES FILE” button.



3.3 Data Recording

To create or change information on a data file, choose the "DATA FILE" button on the main menu. The data file selection screen, similar to the SWIPES value file selection screen, will appear. After selecting the data file, users will then enter the necessary data within the data screen. Users can change or view stored data in the data file by clicking the desired types of material. The program will then show all stored data on this specified material. Users can access the stored information in the data file from inside the component screen by clicking the "DATA FILE" button.

Figure 3.12 Data file selection screen

DATA FILE SELECTION

File Name:

Directories:

☒ c:\
☐ ~mssetup.t
☐ cafe
☐ carpet.cd
☐ coreja~1
☐ corel50
☐ exchange

List Files of Type:

Drives:

Figure 3.13 Data information screen

DATA INFORMATION

PRINT SCREEN

Please enter the following information. After the information is completed, click RECORD button.

Type of material

Arbitrary Value for

Compacted Density	<input type="text" value=".00536"/>	Reuse Value	<input type="text" value=".7"/>
	<input type="radio"/> Lb./in ³ <input type="radio"/> g/cm ³	Recycling Rate Value	<input type="text" value=".6"/>
Fraction Recycled	<input type="text" value=".634"/>	Recycling Content Value	<input type="text" value=".5"/>
Fraction Composted	<input type="text" value="0"/>	Composting Value	<input type="text" value=".2"/>
Fraction Incinerated	<input type="text" value="0"/>	Combustion Value	<input type="text" value=".2"/>
Fraction of Ash after Combustion	<input type="text" value="1"/>		
Fraction Residue after Compost Screen	<input type="text" value="1"/>		

CHAPTER IV

CONCLUSIONS

4.1 Advantages of the Program

In the SWIPES formula, many factors (almost 20) are needed in the calculation for each material. It will take a plenty of time to work by hands. The SWIPES computer program is helpful in minimizing time loss which is important in industries and businesses.

The SWIPES computer program was developed based on the SWIPES model, which was a first frontier model. Therefore, much more can be done to add usefulness to the model. This program is not only useful to reduce time in calculation during packaging design, but also useful in the next period of formula development. Moreover, to support the future changes in the model, the SWIPES computer program was designed in the Visual Basic environment which is very flexible. The way this program was designed allows users to change the formula, variables, information arrays, etc., in the program whenever they need.

Additionally, for the users' conveniences, a material standard data file is supplied with the program. All information in this data file (average number in the US, in 1994) is analysed from the information in Characterization of Municipal Solid Waste in the United

States:1994 Update (EPA, 1994). However, if users want any different information, they can either change or input a specific data by keyboard or create a new data file for their uses.

4.2 Future Development

In order to develop this program, programmers can change all necessary commands by using a Microsoft Visual Basic Software version 3.0 or newer. The computer codes are recorded in "SWIPE.MAK file. They are in alphabetic sequences of forms and objects. The original codes of the program are presented in Appendix A of this paper. Because the SWIPES model might be developed more with the future study, this program should be developed simultaneously, if possible.

As mentioned above, the SWIPES formula and computer program are only the first step in this field, hence, there is not a standard measure of many factors used in the program, especially the arbitrary values. For this reason, Saputo stated in his thesis that "the SWIPES model had to have the flexibility to be explicit to each individual situation". Thus, at present, users are allowed to ponder and weight the important levels of reuse, recycle, compose, and combustion, and then picked a number between 0 to 1 for these values by themselves. As a result of future study, there might be a better method to derive these factors and this should be part of the program development.

LIST OF REFERENCES

LIST OF REFERENCES

Abbott, Donald L., *Packaging Perspectives*, Kendall/Hunt Publishing Co., Dubuque, Iowa (1989)

EPA, *Characterization of Municipal Solid Waste in the United States: 1994 Update*, US Environmental Protection Agency, EPA/530-R-94-042 (November 1994)

EPA, *Characterization of Municipal Solid Waste in the United States: 1990 Update*, US Environmental Protection Agency (1992)

Manypenny, Gerry O., Holmgren, R. Bruce, Beckman, Harold M., Egan, Richard, and McQueen, Gloria, *Glossary of Packaging Terms*, The Packaging Institute, USA, Stamford, Connecticut, Sixth edition, PP 166 (1996)

***Microsoft Visual Basic for Windows 95, Version 3.0*, Microsoft Corporation**

Millard, Peter G., *General Information about Microsoft Visual Basic*, Internet: <http://www.apexsc.com/vb/ftp/vbfaq/vbfaqgen.txt> (1997)

Norton, Peter, *Guide to Visual Basic 4 for Windows 95*, Sam Publishing , Indianapolis, IN (1995)

Perry, Greg, *Visual Basic in 12 Easy lessons*, Sam Publishing , Indianapolis, IN (1995)

Saputo, Brian Peter, *Solid Waste Integrated Packaging Evaluation System*, M.S. Thesis, Michigan State University (1991)

Zak, Diane, *Programming with Microsoft Visual Basic 2.0/3.0 for Windows*, Course Technology, Inc., Cambridge, MA (1995)

ADDITIONAL SOURCES FOR COMPUTER PROGRAMMING

ADDITIONAL SOURCES FOR COMPUTER PROGRAMMING

Socha, John and Hall, Devra, *Teach yourself... Visual Basic 3.0*, MIS:Press, Inc., New York, NY (1994)

Thomas, Zane, Arnson, Robert, and Waite, Mitchell, *Visual Basic How-to*, Second edition, Waite Group Press, Inc., Corte Madera, CA (1993)

APPENDIX A

COMPUTER SOURCE CODE FOR SWIPES

APPENDIX A

COMPUTER SOURCE CODE FOR SWIPES

AboutFrm.Frm

```
Sub CmdReturn_Click ( )
```

```
    WhatFrm.Show
```

```
    AboutFrm.Hide
```

```
End Sub
```

ArbilInfo.Frm

```
Sub CmdOK_Click ( )
```

```
    'Check negative information.
```

```
    IF Val(TxtW.Text) < 0 Or Val(TxtX1.Text) < 0 Or Val(TxtX2.Text) < 0 Or
```

```
    `Val(TxtY.Text) < 0 Or Val(TxtZ.Text) < 0 Then
```

```
        Warn = MsgBox ("Information cannot be a negative number",
```

```
        `MB_IconExclamation, "Please Check")
```

```
    Exit Sub
```

```
End If
```

'Check arbitrary information.

If Val(TxtW.Text) > 1 Or Val(TxtX1.Text) > 1 Or Val(TxtX2.Text) > 1 Or

'Val(TxtY.Text) > 1 Or Val(TxtZ.Text) > 1 Then

Warn = MsgBox ("Arbitrary value cannot be more than one",

'MB_IconExclamation, "Please Check")

Exit Sub

End If

'Assign variables from information array.

W = Val(TxtW.Text)

X1 = Val(TxtX1.Text)

X2 = Val(TxtX2.Text)

Y = Val(TxtY.Text)

Z = Val(TxtZ.Text)

'Record information into information array.

InfoArray(MatCount).W = W

InfoArray(MatCount).X1 = X1

InfoArray(MatCount).X2 = X2

InfoArray(MatCount).Y = Y

InfoArray(MatCount).Z = Z

'Display MatMenuFrm and hide the screen.

ArbInfoFrm.Hide

End Sub

SubForm_Activate ()

'Assign variables on this screen from data file.

TxtW.Text = InfoArray(MatCount).W

TxtX1.Text = InfoArray(MatCount).X1

TxtX2.Text = InfoArray(MatCount).X2

TxtY.Text = InfoArray(MatCount).Y

TxtZ.Text = InfoArray(MatCount).Z

End Sub

Sub Form_Deactivate ()

'Clear old information on the screen.

TxtW.Refresh

TxtX1.Refresh

TxtX2.Refresh

TxtY.Refresh

TxtZ.Refresh

End Sub

Sub MnuFileExit_Click ()

End

End Sub

Sub MnuFileSwipe_Click ()

'Record SWIPES information.

Call SwipeFile

End Sub

Sub MnuInfoMat_Click ()

'Work like OK button.

Call CmdOK_Click

MatInfoFrm.Show

End Sub

Sub MnuInfoPkg_Click ()

'Clear FName variable, protect program from reading a closed file.

If FName <> " " Then

Close #1

FName = " "

End If

'Work like OK button.

Call CmdOK_Click

ComInfoFrm.Show

'Memo how to make the program work.

MsgBox ("Click on of Type of Package to get information")

End Sub

Sub MnuInfoPro_Click ()

'Clear FName variable, protect program from reading a closed file.

If FName <> " " Then

Close #1

FName = " "

End If

'Work like OK button.

Call CmdOK_Click

PPFrm.Show

End Sub

Sub MnuInfoScr_Click ()

'Work like OK button.

Call CmdOK_Click

ScrInfoFrm.Show

End Sub

BInfo.Frm

Sub CmdBack_Click ()

'Display the what to do menu and hide the screen.

WhatFrm.Show

PPFrm.Hide

End Sub

Sub CmdCancle_Click ()

'Choises what the user want to do

**Ques = MsgBox ("Would you like to return to what to do menu? (Yes to return to
'what to do menu, No to clear all information and enter again, or Cancle to keep all
'information and stay in this interface.)", MB_IconQuestion + MB_YesNoCancle, " ")**

Select Case Ques

Case IDYes

WhatFrm.Show

PPFrm.Hide

Case IDNo

TxtPkgName.Text = " "

TxtPkgMan.Text = " "

TxtMassPr.Text = " "

TxtProden.Text = " "

TxtComb.Text = " "

OptGram.Value = False

OptGramPro.Value = False

OptLb.Value = False

OptLbPro.Value = False

Case Else

Exit Sub

End Select

End Sub

Sub CmdNext_Click ()

'Check that all information are positive number.

If Val(TxtMassPr.Text) < 0 Or Val(TxtProden.Text) < 0 Then

Warn = MsgBox("Information cannot be a negative number",

'MB_IconExclamation, "Please check")

Exit Sub

End If

'Check the unit of mass and density of product.

If Convert = Val(" ") Or ConvertPro = Val(" ") Then

Warn = MsgBox ("Please check the units of mass and compacted density

'of product", MB_IconExclamation, "Please check")

Exit Sub

End If

'Assign values of all variables.

ProType = TxtPkgName.Text

```

Man = TxtPkgMan.Text

mPr = Val(TxtMassPr.Text) * Convert

dPr = Val(TxtProden.Text) * ConvertPro

Pcomb = Val(TxtComb.Text)

ProMass = Val(TxtMassPr.Text)

ProDen = Val(TxtProDen.Text)

```

'Display the next screen and hide this screen.

```
ComInfoFrm.Show
```

```
PPFrm.Hide
```

End Sub

Sub Form_Activate ()

'Assign information onto screen.

```
TxtPkgName.Text = ProType
```

```
TxtPkgMan.Text = Man
```

```
TxtMassPr.Text = ProMass
```

```
TxtProDen.Text = ProDen
```

```
TxtComb.Text = Pcomb
```

```
Select Case Left(ProU, 1)
```

```
Case "L"
```

```
Optlb.Value = True
```

```
Case "g"
```

OptGram.Value = True

End Select

Select Case Left(Produ, 1)

Case "L"

OptLbPro.Value = True

Case "g"

OptGramPro.Value = True

End Select

End Sub

Sub OptGram_Click ()

'Assign the converter value.

Convert = 1 / 454

ProU = "g."

End Sub

Sub OptGramPro_Click ()

'Assign the converter value.

ConvertPro = (2.54 ^ 3) / 454

Produ = "g/cm^3"

End Sub

```
Sub OptLb_Click ( )
```

```
    'Assign the converter value.
```

```
        Convert = 1
```

```
        ProU = "Lb."
```

```
End Sub
```

```
Sub OptLbPro_Click ( )
```

```
    'Assign the converter value.
```

```
        ConvertPro = 1
```

```
        Produ = "Lb./in^3"
```

```
End Sub
```

```
ComInfo.Frm
```

```
Sub CmdBack_Click ( )
```

```
    'Display the what to do menu and hide this form.
```

```
        WhatFrm.Show
```

```
        ComInfoFrm.Hide
```

```
End Sub
```

```
Sub CboPkgType_Click ( )
```

```
    'Assign variables from information array to a responding package.
```

```
        PkgType = CboPkgType.Text
```

MatCount = 1

Found = " "

Do While MatCount <= 10

 If CboPkgType.Text = InfoArray(MatCount).pkg Then

 Found = "Yes"

 Exit Do

 End If

 MatCount = MatCount + 1

Loop

If Found = "Yes" Then

 CboTMat.Text = InfoArray(MatCount).Mat

 If Left (InfoArray(MatCount).PkgU, 1) = "g" Then

 OptGramPkg.Value = True

 TxtMassPkg.Text = InfoArray(MatCount).mPkg * 454

 Else

 OptLbPkg.Value = True

 TxtMassPkg.Text = InfoArray(MatCount).mPkg

 End If

 TxtNPro.Text = InfoArray(MatCount).Npro

 TxtNumUse.Text = InfoArray(MatCount).n

End If

End Sub

Sub CmdCancel_Click ()

'Decide where to go.

**Ques = MsgBox ("Would you like to return to what to do menu? (Yes to return to
'what to do menu, No to clear all information and enter again, or Cancel to keep all
'information and stay in this interface.)", MB_IconQuestion + MB_YesNoCancel, " ")**

Select Case Ques

Case IDYes

WhatFrm.Show

CboTMat.Clear

TxtMassPkg.Text = " "

TxtNumUse.Text = " "

OptLbPkg.Value = False

OptGramPkg.Value = False

ComInfoFrm.Hide

Case IDNo

CboPkgType.ListIndex = 0

CboTMat.Clear

TxtMassPkg.Text = " "

TxtNumUse.Text = " "

OptLbPkg.Value = False

OptGramPkg.Value = False

Case Else

Exit Sub

End select

'Protect program from reading a closed file.

If FName <> " " Then

Close #1

FName = " "

End If

End Sub

Sub CmdFile_Click ()

'Protect program from reading a closed file.

If FName <> " " Then

Close #1

FName = " "

End If

SelectForm = 2

FrmGetFile.Show

End Sub

Sub Cmd Next_Click ()

'Check negative information.

If Val(TxtMassPkg.Text) < 0 Val(TxtNumUse.Text) < 0 Then

Warn = MsgBox ("Information cannot be a negative number",

'MB_IconExclamation, "Please Check")

Exit Sub

End If

'Check needed information.

If Val(TxtNumUse.Text) < 1 Then

Warn = MsgBox ("Number of uses should be at least one",

MB_IconExclamation, "Please Check")

Exit Sub

End If

'Check whether units are applied or not.

If (OptLbPkg.Value <> True) and (OptGramPkg.Value <> True) Then

Warn = MsgBox ("Please check the units of package and product mass.",

MB_IconExclamation, "Please Check")

Exit Sub

End If

'Assign variables from screen information.

MPkg = Val(TxtMassPkg.Text) * ConvertPkg

n = Val(TxtNumUse.Text)

Npro = Val(TxtNPro.Text)

'Get material information from data file if the user want.

Lett = 1

Do While Lett <= 20

 Tmat = CboTMat.Text & " "

 If Mid(Tmat, Lett, 1) = " " Then

 SelectMat = Left(CboTMat.Text, (Lett - 1))

 Exit Do

 End If

 Lett = Lett + 1

Loop

If FName <> " " Then

 DataNum = 1

 Get #1, DataNum, DataRec

 Do While Not EOF (1)

 Lett = 1

 Do While Lett <= 20

 If Mid(DataRec.MatType, Lett, 1) = " " Then

 Mat = Left (DataRec.MatType, (Lett - 1))

 Exit Do

 End If

 Lett = Lett + 1

 Loop

 If Ucase (SelectMat) = Ucase (Mat) Then

Exit Do

Else

DataNum = DataNum + 1

DenPkg = Val(" ")

Get #1, DataNum, DataRec

End If

Loop

DenPkg = DataRec.Den

rr = DataRec.rr

c = DataRec.c

i = DataRec.i

a = DataRec.a

e = DataRec.e

W = DataRec.W

X1 = DataRec.X1

X2 = DataRec.X2

Y = DataRec.Y

Z = DataRec.Z

Unit = DataRec.Unit

Close #1

FName = " "

'Check for available materials in the file.

If DenPkg = Val(" ") Then

**Warn = MsgBox ("This type of material is not available on the
selected data file name, Would you like to try another file? (Yes to enter file name, No to
enter all information by keyboard, or Cancel to stay in this interface.)",
MB_IconExclamation + MB_IconYesNoCancel, "Not Available")**

Select Case Warn

Case IDYes

FrmGetFile.Show

Exit Sub

Case IDCANCEL

Exit Sub

End Select

End If

End If

'Record Information in information array.

MatCount = 1

Do While MatCount <= 10

If CboPkgType.Text = InfoArray(MatCount).pkg Then

Found = "Yes"

Exit Do

End If

MatCount = MatCount + 1

Loop

If Found \diamond "Yes" Then

MatCount = 1

Do While MatCount \leq 10

If InfoArray(MatCount).pkg = " " Then

Exit Do

End If

MatCount = MatCount + 1

Loop

InfoArray(MatCount).pkg = CboPkgType.Text

InfoArray(MatCount).Mat = CboTMat.Text

InfoArray(MatCount).mPkg = mPkg

InfoArray(MatCount).PkgU = PkgU

InfoArray(MatCount).n = n

InfoArray(MatCount).NPro = NPro

InfoArray(MatCount).dPkg = DenPkg

InfoArray(MatCount).Pkgdu= Unit

InfoArray(MatCount).rr = rr

InfoArray(MatCount).c = c

InfoArray(MatCount).i = i

InfoArray(MatCount).a = a

InfoArray(MatCount).e = e

InfoArray(MatCount).W = W

InfoArray(MatCount).X1 = X1

InfoArray(MatCount).X2= X2

InfoArray(MatCount).Y = Y

InfoArray(MatCount).Z = Z

End If

Mat = CboTMat.Text

'Display material menu screen.

MatMenuFrm.Show

ComInfoFrm.Hide

End Sub

Sub Form_Activate ()

CboPkgType.Clear

'Add items to the type of packaging combo box.

MatNum = 1

If InfoArray(MatNum).pkg = " " or AddMat = "Yes" Then

CboPkgType.AddItem "Bag"

CboPkgType.AddItem "Bottle"

CboPkgType.AddItem "Box"

CboPkgType.AddItem "Can"

CboPkgType.AddItem "Carton"

```

CboPkgType.AddItem "Pallet"
CboPkgType.AddItem "Pouch"
CboPkgType.AddItem "Tube"
CboPkgType.AddItem "Wrapper"
CboPkgType.AddItem "Other"

```

```
Else
```

```
Do While InfoArray(MatNum).pkg <> ""
```

```
    CboPkgType.AddItem InfoArray(MatNum).pkg
```

```
    MatNum = MatNum + 1
```

```
Loop
```

```
End If
```

```
CboPkgType.ListIndex = 0
```

```
'Set default values.
```

```
TxtMassPkg.Text = ""
```

```
OptGramPkg.Value = False
```

```
OptLbPkg.Value = False
```

```
TxtNumUse.Text = ""
```

```
TxtNPro.Text = ""
```

```
End Sub
```

```
Sub MnuFileExit_Click ( )
```

```
End
```

End Sub

SubMnuFileValue_Click ()

Call SwipeFile

End Sub

SubInfoPro_Click ()

Call CmdNext_Click

PPFrm.Show

End Sub

SubOptGramPkg_Click ()

'Assign the converter value.

ConvertPkg = 1 / 454

PkgU = "g."

End Sub

Sub OptLbPkg_Click ()

'Assign the converter value.

ConvertPkg = 1

PkgU = "Lb."

End Sub

DataFrm.Frm

Sub CboTMat_Click ()

‘Clear old information on screen.

 TxtDen.Refresh

 TxtRRate.Refresh

 TxtCRate.Refresh

 TxtIRate.Refresh

 TxtW.Refresh

 TxtX1.Refresh

 TxtX2.Refresh

 TxtY.Refresh

 TxtZ.Refresh

 TxtAsh.Refresh

 TxtRes.Refresh

 OptGramPkg.Value = False

 OptLbPkg.Value = False

‘Get informaiton from a file if it is opened.

 If FName <> “ “ Then

 DataNum = 1

 Get #1, DataNum, DataRec

 Do While Not EOF (1)

Lett = 1

Do While Lett <= 20

If Mid(DataRec.MatType, Lett, 1) = " " Then

Mat = Left(DataRec.MatType, (Lett - 1))

Lett = 21

End If

Lett = Lett + 1

Loop

Lett = 1

Do While Lett <= 20

If Mid(CboTMat.Text, Lett, 1) = " " Then

SelectMat = Left(CboTMat.Text, (Lett - 1))

Lett = 21

End IF

Lett = Lett + 1

Loop

If Ucase(SelectMat) = Ucase(Mat) Then

TxtRRate.Text = DataRec.Rrate

TxtCRate.Text = DataRec.CRate

TxtIRate.Text = DataRec.IRate

TxtW.Text = DataRec.W

TxtX1.Text = DataRec.X1

```
TxtX2.Text = DataRec.X2
TxtY.Text = DataRec.Y
TxtZ.Text = DataRec.Z
TxtAsh.Text = DataRec.Ash
TxtRes.Text = DataRec.Res
Unit = DataRec.Unit
If Unit = "g/cm^3" Then
    OptGramPkg.Value = True
    TxtDen.Text = DataRec.Den * 454 / (2.54^3)
Else
    OptLbPkg.Value = True
    TxtDen.Text = DataRec.Den
End If
Exit Do
Else
    DataNum = DataNum + 1
    Get #1, DataNum, DataRec
End If
Loop
End If
End Sub
```

Sub CmdCancel_Click ()

'Show the what to do form and hide this form/

Close #1

FName = " "

**Ques = MsgBox (Would you like to get a different file?", MB_IconQuestion +
MB_YesNo, "Different File")**

Select Case Ques

Case IDYes

FrmGetFile.Show

DataFrm.Hide

Case IDNo

WhatFrm.Show

DataFrm.Hide

End Select

End Sub

Sub CmdRecord_Click ()

'Make sure one of the unit option boxes is checked.

If Unit = " " Then

**Warn = MsgBox ("Please Check the Units of package and product
compacted density.", MB_IconExclamation, "Please check")**

Exit Sub

End If

'Check negative informaiton.

If Val(TxtDen.Text) < 0 Or Val(TxtRRate.Text) < 0 Or Val(TxtCRate.Text) < 0
 `Or Val(TxtW.Text) < 0 Or Val(TxtX1.Text) < 0 Or Val(TxtX2.Text) < 0 Or
 `Val(TxtY.Text) < 0 Or Val(TxtZ.Text) < 0 Or Val(TxtAsh.Text) < 0 Or
 `Val(TxtRes.Text) < 0 Then

Warn = MsgBox ("Information cannot be a negative number",
 `MB_IconExclamation, "Please Check")

Exit Sub

End If

'Check arbitrary information.

If Val(TxtW.Text) > 1 Or Val(TxtX1.Text) > 1 Or Val(TxtX2.Text) > 1 Or
 `Val(TxtY.Text) > 1 Or Val(TxtZ.Text) > 1 Then

Warn = MsgBox ("Arbitrary values cannot be more than one",
 `MB_IconExclamtion, "Please Check")

Exit Sub

End If

'Check fraction information.

If Val(TxtAsh.Text) > 1 Or TxtRes.Text) > 1 Then

Warn = MsgBox (Please enter all fraction number as fraction (less than 1,
 `without % sign)", MB_IconExclamation, "Please Check")

Exit Sub

End If

'Write the new information onto the data file, change the old information if the same
'material is already recorded or write to the bottom of file if it is new material.

DataNum = 1

Get #1, DataNum, DataRec

Do While Not EOF(1)

 If DataRec.MatType <> " " Then

 If Ucase(CboTMat.Text) = Ucase DataRec.MatType) Then

 Found = "Yes"

 Exit Do

 End If

 Else

 Exit Do

 End If

 DataNum = DataNum + 1

 Get #1, DataNum, DataRec

Loop

DataRec.MatType = CboTMat.Text

If Left(Unit, 1) = "g" Then

 DataRec.Den = Val(TxtDen.Text) * (2.54^3) / 454

Else

 DataRec.Den = Val(TxtDen.Text)

End If

DataRec.RRate = Val(TxtRRate.Text)

DataRec.CRate = Val(TxtCRate.Text)

DataRec.IRate = Val(TxtIRate.Text)

DataRec.W = Val(TxtW.Text)

DataRec.X1 = Val(TxtX1.Text)

DataRec.X2 = Val(TxtX2.Text)

DataRec.Y = Val(TxtY.Text)

DataRec.Z = Val(TxtZ.Text)

DataRec.Ash = Val(TxtAsh.Text)

DataRec.Res = Val(TxtRes.Text)

DataRec.Unit = Unit

Put #1, DataNum, DataRec

Close #1

‘Whether the user want to record more, where.

Record = MsgBox “Would you like to record more in the same data file?”,

‘MB_IconQuestion + MB_YesNo, “Same File”)

Select Case Record

Case IDYes

Open FName For Random As #1 Len = Len(DataRec)

Case IDNo

Drecord = MsgBox (“More in a different file?”, MB_IconQuestion

‘+ MB_YesNo, “Different File”)

Select Case Drecord

Case ID Yes

FrmGetFile.Show

Case IDNo

WhatFrm.Show

DataFrm.Hide

End Select

End Select

End Sub

Sub Form_Activate ()

‘Clear old information.

OptGramPkg.Value = False

OptLbPkg.Value = False

TxtAsh.Refresh

TxtCRate.Refresh

TxtDen.Refresh

TxtIRate.Refresh

CboTMat.Clear

TxtRes.Refresh

TxtRRate.Refresh

TxtW.Refresh

TxtX1.Refresh

TxtX2.Refresh

TxtY.Refresh

TxtZ.Refresh

'Add items to CboTMat.

If FName <> " " Then

DataNum = 1

Get #1, DataNum, DataRec

Do While Not EOF(1)

CboTMat.AddItem DataRec.MatType

DataNum = DataNum + 1

Get #1, DataNum, DataRec

Loop

CboTMat.ListIndex = 0

End If

End Sub

Sub MnuPrint_Click ()

CmdCancel.Visible = False

CmdRecord.Visible = False

PrintForm

CmdCancel.Visible = True

CmdRecord.Visible = True

End Sub

Sub OptGramPkg_Click ()

'Assign unit variable.

Unit = "g/cm^3"

End Sub

Sub OptLbPkg_Click ()

'Assign unit variable.

Unit = "lb/in^3"

End Sub

DoWhat.Frm

Sub CmdData_Click ()

'Display the data form and hide this form.

SelectForm = 1

FrmGetFile.LstFileType.Clear

FrmGetFile.LstFileType.AddItem "SWIPES Data Files (*.swp)"

FrmGetFile.LstFileType.AddItem "All File (*.*)"

FrmGetFile.LstFileType.ListIndex = 0

FrmGetFile.TxtFiles.Text = "*.swp"

FrmGetFile.Caption = "Data File Selection"

FrmGetFile.Show

End Sub

Sub CmdExit_Click ()

End

End Sub

Sub CmdSwipeFile_Click ()

'Protect program from reading a closed file.

If FName <> " " Then

Close #1

FName = " "

End If

SelectForm = 3

FrmGetFile.LstFileType.Clear

FrmGetFile.LstFileType.AddItem "SWIPES Value Files (*.val)"

FrmGetFile.LstFileType.AddItem "All File (*.*)"

FrmGetFile.LstFileType.ListIndex = 0

FrmGetFile.TxtFiles.Text = "*.val"

FrmGetFile.Caption = "SWIPES Value File Selection"

FrmGetFile.Show

End Sub

```
Sub CmdAbout_Click ( )
```

```
    AboutFrm.Show
```

```
    WhatFrm.Hide
```

```
End Sub
```

```
Sub Form_Activate ( )
```

```
    'Clear old information in infoArray.
```

```
    For MatCount = 1 To 10
```

```
        InfoArray(MatCount).pkg = " "
```

```
        InfoArray(MatCount).Mat = " "
```

```
        InfoArray(MatCount).mPkg = 0
```

```
        InfoArray(MatCount).PkgU = " "
```

```
        InfoArray(MatCount).n = 0
```

```
        InfoArray(MatCount).dPkg = 0
```

```
        InfoArray(MatCount).Pkgdu = " "
```

```
        InfoArray(MatCount).rr = 0
```

```
        InfoArray(MatCount).c = 0
```

```
        InfoArray(MatCount).i = 0
```

```
        InfoArray(MatCount).a = 0
```

```
        InfoArray(MatCount).e = 0
```

```
        InfoArray(MatCount).ms = 0
```

InfoArray(MatCount).Scrapu = “ “

InfoArray(MatCount).f = 0

InfoArray(MatCount).b = 0

InfoArray(MatCount).rc = 0

InfoArray(MatCount).W = 0

InfoArray(MatCount).X1 = 0

InfoArray(MatCount).X2= 0

InfoArray(MatCount).Y = 0

InfoArray(MatCount).Z = 0

InfoArray(MatCount).NPro = 0

MatSwipeArray(MatCount).Mat = “ “

MatSwipeArray(MatCount).VCom = 0

MatSwipeArray(MatCount).MScr = 0

MatSwipeArray(MatCount).Reuse = 0

MatSwipeArray(MatCount).Recy = 0

MatSwipeArray(MatCount).Comp = 0

MatSwipeArray(MatCount).Comb = 0

MatSwipeArray(MatCount).ProEf = 0

MatSwipeArray(MatCount).ScrFac = 0

MatSwipeArray(MatCount).PFac = 0

MatSwipeArray(MatCount).Swipe = 0

Next MatCount

End Sub

Sub SwipeCmd_Click ()

PPFrm.Show

WhatFrm.Hide

End Sub

FrmGetFile.Frm

Sub CmdCancel_Click ()

'Hide data file selection form.

TxtFiles.Refresh

FrmGetFile.Hide

End Sub

Sub CmdOK_Click ()

'Assign data file name.

Select Case SelectForm

Case 1

If Right (TxtFiles.Text, 3) = "swp" Then

FName = TxtFiles.Text

Else

FName = TxtFiles.Text & ".swp"

End If

Case 4

If Right (TxtFiles.Text, 3) = ".val" Then

FName = TxtFiles.Text

Else

FName = TxtFiles.Text & ".val"

End If

Case Else

FName = TxtFiles.Text

End Select

If Right\$(FilFile.Path, 1) = "\" Then

FName = FilFile.Path & FName

Else

FName = FilFile.Path & "\" & FName

End If

ComInfoFrm.CboTMat.Clear

'Check If the FName is available or not.

If SelectForm <> 4 Or 1 Then

If Dir\$(FName) = "" Then

MsgBox "The selected file name does not exist, please check the
'name and enter again."

Exit Sub

End If

End If

'Open the data file if it is available.

Open FName For Random As #1 Len = Len(DataRec)

TxtFiles.Refresh

Select Case SelectForm

Case 1

DataFrm.Show

Case 2

ComInfoFrm.CboTMat.Clear

DataNum = 1

'Compare and get information from data file.

Get #1, DataNum, DataRec

Do While Not EOF(1)

ComInfoFrm.CboTMat.AddItem DataRec.MatType

DataNum = DataNum + 1

Get #1, DataNum, DataRec

Loop

ComInfoFrm.CboTMat.ListIndex = 0

Case 3

DataNum = 1

Get #1, DataNum, DataRec

ProType = DataRec.ProType

Man = DataRec.Man

ProMass = DataRec.ProMass

ProDen = DataRec.ProDen

PComb = DataRec.PComb

ProU = DataRec.ProU

Produ = DataRec.Produ

Do While Not EOF (1)

InfoArray(DataNum).pkg = DataRec.pkg

InfoArray(DataNum).Mat = DataRec.Mat

InfoArray(DataNum).mPkg = DataRec.mPkg

InfoArray(DataNum).PkgU = DataRec.PkgU

InfoArray(DataNum).n = DataRec.n

InfoArray(DataNum).dPkg = DataRec.dPkg

InfoArray(DataNum).Pkgdu = DataRec.Pkgdu

InfoArray(DataNum).rr = DataRec.RRate

InfoArray(DataNum).c= DataRec.CRate

InfoArray(DataNum).i = DataRec.IRate

InfoArray(DataNum).a = DataRec.Ash

InfoArray(DataNum).e = DataRec.Res

InfoArray(DataNum).ms = DataRec.ms

InfoArray(DataNum).Scrapu = DataRec.Scrapu

```

InfoArray(DataNum).f = DataRec.f
InfoArray(DataNum).b = DataRec.b
InfoArray(DataNum).W = DataRec.W
InfoArray(DataNum).X1 = DataRec.X1
InfoArray(DataNum).X2 = DataRec.X2
InfoArray(DataNum).Y = DataRec.Y
InfoArray(DataNum).Z = DataRec.Z
InfoArray(DataNum).NPro = DataRec.NPro
DataNum = DataNum + 1
Get #1, DataNum, DataRec

```

Loop

Close #1

PPFrm.Show

LstFileType.Clear

LstFileType.AddItem "SWIPES Data Files (*.swp)"

LstFileType.AddItem "All File (*.*)"

LstFileType.ListIndex = 0

FrmGetFile.Caption = "Data File Selection"

FName = " "

Case 4

'Record SWIPES information onto a SWIPES value file.

MatC = 1

Do While MatC <= MatCount

DataNum = 1

Get #1, DataNum, DataRec

DataRec.ProType = ProType

DataRec.Man = Man

DataRec.ProMass = ProMass

DataRec.ProDen = ProDen

DataRec.PComb= PComb

DataRec.ProU= ProU

DataRec.Produ = Produ

Do While Not EOF(1)

If DataRec.pkg = “ “ Or DataRec.pkg = InfoArray(MatC).pkg Then

Exit Do

End If

DataNum = DataNum + 1

Get #1, DataNum, DataRec

Loop

DateRec.Pkg = InfoArray(MatC).Pkg

DateRec.MatType = InfoArray(MatC).MatType

DateRec.mPkg = InfoArray(MatC).mPkg

DateRec.PkgU = InfoArray(MatC).PkgU

DateRec.n = InfoArray(MatC).n

```

DateRec.Den = InfoArray(MatC).dPkg
DateRec.Unit = InfoArray(MatC).Pkgdu
DateRec.RRate = InfoArray(MatC).rr
DateRec.CRate = InfoArray(MatC).c
DateRec.IRate = InfoArray(MatC).i
DateRec.Ash = InfoArray(MatC).a
DateRec.Res = InfoArray(MatC).e
DateRec.ms = InfoArray(MatC).ms
DateRec.Scrapu = InfoArray(MatC).Scrapu
DateRec.f = InfoArray(MatC).f
DateRec.b = InfoArray(MatC).b
DateRec.rc = InfoArray(MatC).rc
DateRec.W = InfoArray(MatC).W
DateRec.X1 = InfoArray(MatC).X1
DateRec.X2 = InfoArray(MatC).X2
DateRec.Y = InfoArray(MatC).Y
DateRec.Z = InfoArray(MatC).z
DateRec.NPro = InfoArray(MatC).Npro
Put #1, DataNum, DateRec
MatC = MatC + 1

```

Loop

Close #1

LstFileType.Clear

LstFileType.AddItem "SWIPES Data Files (*.swp)"

LstFileType.AddItem "All File (*.*)"

LstFileType.ListIndex = 0

FrmGetFile.Caption = "Data File Selection"

End Select

'Hide data file selection form.

FrmGetFile.Hide

End Sub

Sub DirList_Change ()

'Change the file list with the selection of the directory.

FilFile.Path = DirList.Path

If FilFile.ListCount > 0 Then

FilFile.ListIndex = 0

End If

LblDirs.Caption = DirList.Path

End Sub

Sub DrvDrive_Change ()

'Change the directory path with the change of drive selected.

On Error GoTo ErrHandler

DirList.Path = DrvDrive.Drive

Exit Sub

ErrorHandler:

Const MB_RetryCancel = 5, MB_OK = 0

Const MB_IconExclamation = 48, MB_IconStop = 16

Const IDRetry = 4, IDCancel = 2

Select Case Err

Case Device_Unavailable, Disk_Not_Ready

**Warn = MsgBox("Please Check the disk.", MB_IconExclamation
' + MB_RetryCancel, "Error")**

If Warn = IDRetry Then

Resume

Else

DrvDrive.Drive = DirList.Path

**Resume Next 'Resume with DirList.Path =
'DrvDrive.Drive statement**

End If

Case Else

**Warn = MsgBox ("Unrecoverable error:" & Error\$, MB_IconStop
' + MB_OK, "Error")**

Resume Next 'Resume with Exit Sub Statement

End Select

End Sub

Sub FilFile_Click ()

TxtFiles.Text = FilFile.List (FilFile.ListIndex)

End Sub

Sub FilFile_DblClick ()

Call CmdOK_Click

End Sub

Sub Form_Activate ()

‘Set default values.

DrvDrive.Drive = “c:”

DirList.Path = “c:\” ‘Root directory

LblDirs.Caption = DirList.Path

‘Select the first file in the list of files.

FilFile.Refresh

If FilFile.ListCount > 0 then

FilFile.ListIndex = 0

End If

End Sub

Sub LstFileType_Click ()

'Change the file list pattern with the selection of the file type.

Select Case LstFileType.Text

Case "Swipes Data Files (*.swp)"

FilFile.Pattern = "*.swp"

Case "Swipes Value Files (*.val)"

FilFile.Pattern = "*.val"

Case "All File (*.*)"

FilFile.Pattern = "*.*)"

End Select

End Sub

MatInfo.Frm

Sub CmdOK_Click ()

'Check whether any density value is Zero, warn if they are.

If Val(TxtPkgDen.Text) = 0 Then

Warn= MsgBox ("The density cannot be 0 (Zero)", MB_IconExclamation,

"Please Check")

Exit Sub

End If

'Check whether units are applied or not.

If Convert = Val(" ") Then

```

Warn = MsgBox ("Please check the units of package and product mass.",
'MB_IconExclamation, "Please check")

Exit Sub

End If

'Check negative information.

If Val(TxtPkgDen.Text) < 0 Or Val (TxtRecRate.Text) < 0 Or
'Val(TxtFracAsh.Text) < 0 Or Val(TxtIncRate.Text) < 0 Or Val(TxtRes.Text < 0 Then

Warn = MsgBox("Information cannot be a negative number",
'MB_IconExclamation, "Please Check")

Exit Sub

End If

'Check fraction information.

If Val(TxtFracAsh.Text) > 1 Or Val(TxtRes.text) > 1 Then

Warn = MsgBox("Please enter all fraction number as fraction (less than 1,
'without % sign)", MB_IconExclamation, "Please Check")

Exit Sub

End If

'Assign variables from those text boxes on the screen.

dPkg = Val(TxtPkgDen.Text) * Convert

π = Val(TxtRecRate.Text)

c = Val(TxtCompRate.Text)

i = Val(TxtIncRate.Text)

```

a = Val(TxtFracAsh.Text)

e = Val(TxtRes.Text)

'Record information into information array.

InfoArray(MatCount).dPkg = dPkg

InfoArray(MatCount).Pkgdu = Pkgdu

InfoArray(MatCount).rr = rr

InfoArray(MatCount).c = c

InfoArray(MatCount).i = i

InfoArray(MatCount).a = a

InfoArray(MatCount).e= e

'Display MatMenufrm and hide this form.

MatInfoFrm.Hide

End Sub

Sub Form_Activate ()

'Clear old information.

TxtPkgDen.Refresh

TxtRecRate.Refresh

TxtCompRate.Refresh

TxtIncRate.Refresh

TxtFracAsh.Refresh

TxtRes.Refresh

OptLbPkg.Value = False

OptGramPkg.Value = False

'Assign variables from infoarray if available/

Unit = InfoArray(MatCount).Pkgdu

Select Case Left(Ucase(Unit), 1)

Case " L"

OptLbPkg.Value = True

TxtPkgDen.Text = InfoArray(MatCount.dPkg

Case " G"

OptGramPkg.Value = True

TxtPkgDen.Text = InfoArray(MatCount.dPkg * 454 / (2.54^3)

End Select

TxtTecRate.Text = InfoArray(MatCount).rr

TxtCompRate.Text = InfoArray(MatCount).c

TxtIncRate.Text = InfoArray(MatCount).i

TxtFracAsh.Text = InfoArray(MatCount).a

TxtRes.Text = InfoArray(MatCount).e

End Sub

Sub MnuFileExit_Click ()

End

End Sub

```
Sub MnuFileSwipe_Click ( )
```

```
    Call SwipeFile
```

```
End Sub
```

```
Sub MnuInfoArb_Click ( )
```

```
    Call CmdOK_Click
```

```
    ArblInfoFrm.show
```

```
End Sub
```

```
Sub MnuInfoPkg_Click ( )
```

```
    If FName <> " " Then
```

```
        Close #1
```

```
        FName = " "
```

```
    End If
```

```
    Call CmdOK_Click
```

```
    ComInfoFrm.Show
```

```
    MsgBox "Click ont of Type of Packge to get information"
```

```
End Sub
```

```
Sub MnuInfoPro_Click ( )
```

```
    If FName <> " " Then
```

Close #1

FName = “ “

End If

Call CmdOK_Click

PPFrm.Show

End Sub

Sub MnuInfoScr_Click ()

Call CmdOK_Click

ScrInfoFrm.Show

End Sub

Sub OptGramPkg_Click ()

‘Assign the converter value.

Convert = (2.54^3) / 454

Pkgdu = “g./cm^3”

End Sub

Sub OptLbPkg_Click ()

‘Assign the converter value.

Convert = 1

Pkgdu = “Lb./in^3”

End Sub

MatMenu.Frm

Sub Command1_Click ()

'Display MatInfoFrm and hide this screen.

MatInfoFrm.Show

End Sub

Sub Command2_Click ()

'Display ScrInfoFrm and hide the screen.

ScrInfoFrm.Show

End Sub

Sub Command3_Click ()

'Display ArbInfoFrm and hide the screen.

ArbInfoFrm.Show

End Sub

Sub Command4_Click ()

'Display the material Swipe value form and hide this form.

MatSwipeFrm.Show

End Sub

Sub Command5_Click ()

“Check necessary information.

If InfoArray(MatCount).dPkg = 0 Then

Warn = MsgBox(“The density value cannot be 0(Zero)”,

MB_IconExclamation, “Please Check”)

MatInfoFrm.Show

MatMenuFrm.Hide

Exit Sub

End If

“Check the previous material SWIPES values and record those values in MatSwipeArray.

If Vcom = Val(“”) Then

Call Calculation

Call SwipeRec

End If

“Display the component information form and hide this form.

FName = “”

ComInfoFrm.Show

End Sub

Sub Command6_Click ()

“Check necessary information.

If InfoArray(MatCount).dPkg = 0 Then

```

Warn = MsgBox("The density value cannot be 0(Zero)",
'MB_IconExclamation, "Please Check")

MatInfoFrm.Show

MatMenuFrm.Hide

Exit Sub

End If

If Vcom = Val(" ") Then

    Call Calculation

    Call SwipeRec

End If

"Clear old information in array.

For MatNum = 1 To 10

    InfoArray(MatNum).pkg = " "
    InfoArray(MatNum).Mat = " "
    InfoArray(MatNum).mPkg = 0
    InfoArray(MatNum).PkgU = " "
    InfoArray(MatNum).n = 0
    InfoArray(MatNum).dPkg = 0
    InfoArray(MatNum).Pkgdu = " "
    InfoArray(MatNum).rr = 0
    InfoArray(MatNum).c = 0
    InfoArray(MatNum).i = 0

```

```

InfoArray(MatNum).a = 0
InfoArray(MatNum).e = 0
InfoArray(MatNum).ms = 0
InfoArray(MatNum).Scrapu = " "
InfoArray(MatNum).f = 0
InfoArray(MatNum).b = 0
InfoArray(MatNum).rc = 0
InfoArray(MatNum).W = 0
InfoArray(MatNum).X1 = 0
InfoArray(MatNum).X2 = 0
InfoArray(MatNum).Y = 0
InfoArray(MatNum).Z = 0
InfoArray(MatNum).NPro = 0

```

```
Next MatNum
```

```
WhatFrm.Show
```

```
MatMenuFrm.Hide
```

```
End Sub
```

```
Sub Command7_Click ( )
```

```
‘Check necessary information.
```

```
    If InfoArray(MatCount).dPkg = 0 Then
```

```
        Warn = MsgBox("The density value cannot be 0 (Zero)",
```

```
        ‘MB_IconExclamation, "Please Check")
```

```

        MatInfoFrm.Show

        MatMenuFrm.Hide

        Exit Sub

    End If

    If Vcom = Val(" ") Then

        Call Calculation

        Call SwipeRec

    End If

    ComInfoFrm.Show

    MsgBox("Click one of Type of Package to get informaiton")

End Sub


Sub Command8_Click ( )

'Check necessary information.

    If InfoArray(MatCount).dPkg = 0 Then

        Warn = MsgBox("The density value cannot be 0 (Zero)",

'MB_IconExclamation, "Please Check")

        MatInfoFrm.Show

        MatMenuFrm.Hide

        Exit Sub

    End If

    If Vcom = Val(" ") Then

```

Call Calculation

Call SwipeRec

End If

AddMat = "Yes"

ComInfoFrm.Show

MatMenuFrm.Hide

End Sub

Sub Form_Active ()

Vcom = Val(" ")

End Sub

MatSwipe.Frm

Sub CmdReturn_Click ()

'Display the material menu form and hide this form.

MatMenuFrm.Show

MatSwipeFrm.Hide

End Sub

Sub Form_Active ()

'Clear old information.

LstMat.Clear

LblComb.Caption = “ “

LblComp.Caption = “ “

LblMScr.Caption = “ “

LblPFac.Caption = “ “

LblRecy.Caption = “ “

LblReuse.Caption = “ “

LblScrFac.Caption = “ “

LblProEf.Caption = “ “

LblVCom.Caption = “ “

LblSwipe.Caption = “ “

OptCm.Value = False

OptIn.Value = False

‘Calculate SWIPES value and record values in MatSwipeArray for current material.

Call Calculation

Call SwipeRec

‘Add items to material list box.

MatNum =1

Do While MatSwipeArray(MatNum).Mat < “ “

LstMat.AddItem MatSwipeArray(MatNum).Mat

LstMat.ItemData(LstMat.NewIndex) = MatNum

MatNum = MatNum + 1

Loop

LstMat.AddItem "Total SWIPES Value"

LstMat.ListIndex = MatNum - 2

End Sub

Sub LstMat_Click ()

'Assign Variables on screen for selected material with data in MatSwipeArray.

MatNum = 1

TSwipe = 0

Do While MatSwipeArray(MatNum).Mat <> ""

TSwipe = MatSwipeArray(MatNum).Swipe + TSwipe

MatNum = MatNum + 1

Loop

If LstMat.Text = "Total SWIPES Value" Then

TSwipeFrm.Show

Else

Selection = 1 + LstMat.ListIndex

VCom = MatSwipeArray(Selection).VCom

MScr = MatSwipeArray(Selection).MScr

Reuse = MatSwipeArray(Selection).Reuse

Recy = MatSwipeArray(Selection).Recy

Comp = MatSwipeArray(Selection).Comp

Comb = MatSwipeArray(Selection).Comb


```

ProEf = MatSwipeArray(Selection).ProEf
ScrFac = MatSwipeArray(Selection).ScrFac
PFac = MatSwipeArray(Selection).PFac
Swipe = MatSwipeArray(Selection).Swipe

```

```
End If
```

'Clear all labels before get new values.

```

LblComb.Caption = " "
LblComp.Caption = " "
LblMScr.Caption = " "
LblPFac.Caption = " "
LblRecy.Caption = " "
LblReuse.Caption = " "
LblScrFac.Caption = " "
LblProEf.Caption = " "
LblVCom.Caption = " "
LblSwipe.Caption = " "
OptCm.Value = False
OptIn.Value = False

```

```
End Sub
```

```
Sub MnuFileExit_Click ()
```

```
End
```

End Sub

Sub MnuFileSwipe_Click ()

Call SwipeFile

End Sub

Sub MnuInfoArb_Click ()

ArbInfoFrm.Show

MatSwipeFrm.Hide

End Sub

Sub InfoMat_Click ()

MatInfoFrm.Show

MatSwipeFrm.Hide

End Sub

Sub MnuInfoPkg_Click ()

ComInfoFrm.Show

MatSwipeFrm.Hide

MsgBox "Click ont of Type of Packge to get information"

End Sub

Sub MnuInfoPro_Click ()

PPFrm.Show

MatSwipeFrm.Hide

End Sub

Sub MnuInfoScr_Click ()

ScrInfoFrm.Show

MatSwipeFrm.Hide

End Sub

Sub MnuPrintAll_Click ()

Call Ppro

Call Pmat

Call Pswipe

Printer.Print Tab(10); "***"**

*******"**

Printer.NewPage

End Sub

Sub MnuPrintInfo_Click ()

Call Ppro

Call Pmat

```

Printer.Print Tab(10); "*****
*****"

```

```

Printer.NewPage

End Sub

```

```

Sub MnuPrintResult_Click 9 )

```

```

    Call Ppro

    Call Pswipe

    Printer.Print Tab(10); "*****
*****"

```

```

Printer.NewPage

End Sub

```

```

Sub OptCm_Click ( )

```

```

'Assign the converter value.

```

```

    Convert = 2.54^3

```

```

'Show calculation results.

```

```

    LblVCom.Caption = VCom * Convert

```

```

    LblMScr.Caption = MScr

```

```

    LblReuse.Caption = Reuse

```

```

    LblRecy.Caption = Recy

```

```

    LblComp.Caption = Comp

```

LblComb.Caption = Comb

LblProEf.Caption = ProEf

LblScrFac.Caption = ScrFac * Convert

LblPFac.Caption = PFac * Convert

LblSwipe.Caption = Swipe * Convert

MatSwipeArray(MatNum).Convert = Convert

End Sub

Sub OptIn_Click ()

'Assign the converter value.

Convert = 1

'Show calculation results.

LblVCom.Caption = VCom * Convert

LblMScr.Caption = MScr

LblReuse.Caption = Reuse

LblRecy.Caption = Recy

LblComp.Caption = Comp

LblComb.Caption = Comb

LblProEf.Caption = ProEf

LblScrFac.Caption = ScrFac * Convert

LblPFac.Caption = PFac * Convert

LblSwipe.Caption = Swipe * Convert

MatSwipeArray(MatNum).Convert = Convert

End Sub

ScrInfo.Frm

SubCmdOK_Click ()

'Make sure one of the unit option boxes is checked.

**If Val(TxtMassScr.Text) <> 0 And OptLb.Value <> True And (OptGram.Value
'<> True) Then**

**Warn = MsgBox("Please check the Units of package and product
'compacted density.", MB_IconExclamation, "Please Check")**

Exit Sub

End If

'Check negative information.

**If Val(TxtFracScr.Text) < 0 Or Val(TxtMassScr.Text) < 0 Or
'Val(TxtProDam.Text) < 0 Or Val(TxtRecCon.Text) < 0 Then**

**Warn = MsgBox("Information cannot be a negative number.",
MB_IconExclamation, "Please Check")**

Exit Sub

End If

'Check fraction information.

**If Val(TxtFracScr.Text) > 1 Or Val(TxtProDam.Text) > 1 Or
'Val(TxtRecCon.Text) > 1 Then**

```
Warn = MsgBox("Please enter all fraction number as fraction (less than 1,
'without % Sign).",MB_IconExclamaiton, "Please Check")
```

```
Exit Sub
```

```
End If
```

```
'Check necessary information.
```

```
If Val(TxtProDam.Text) <> InfoArray(1).b And InfoArray(1).b <> 0 Then
```

```
Warn = MsgBox("Fraction of product damage in distribution should be
'the same value for every package. Would you like to change the number?",
'MB_IconQuestion + MB_YesNo, "")
```

```
Select Case Warn
```

```
Case IDYes
```

```
InfoArray(1).b = Val(TxtProDam.Text)
```

```
b = Val(TxtProDam.Text)
```

```
Case IDNo
```

```
B = InfoArray(1).b
```

```
InfoArray(MatCount).b = InfoArray(1).b
```

```
End Select
```

```
End If
```

```
'Assign variables with informaiton from the screen.
```

```
b = Val(TxtProDam.Text)
```

```
ms = Val(TxtMassScr.Text) * Convert
```

```
f = Val(TxtFracScr.Text)
```

rc = Val(TxtRecCon)

'Record information into information array.

If OptLb.Value = True Then

InfoArray(MatCount).Scrapu = "Lb."

Else

If OptGram.Value = True Then

InfoArray(MatCount).Scrapu = "g."

End If

End If

If InfoArray(1).b = 0 Then

InfoArray(1).b = Val(TxtProDam.Text)

End If

InfoArray(MatCount).b = Val(TxtProDam.Text)

InfoArray(MatCount).ms = ms

InfoArray(MatCount).f = f

InfoArray(MatCount).rc = rc

'Display MatMenuFrm and hide the screen.

ScrInfoFrm.Hide

End Sub

Sub Form_Activate ()

TxtFracScr.Refresh


```

TxtMassScr.Refresh

TxtProDam.Refresh

TxtRecCon.Refresh

OptGram.Val = False

OptLb.Val = False

Select Case Left(InfoArray(MatCount).Scrapu, 1)

    Case "g"

        OptGram.Value = True

        TxtMassScr.Text = InfoArray(MatCount).ms * 454

    Case "L"

        OptLb.Value = True

        TxtMassScr.Text = InfoArray(MatCount).ms

End Select

TxtFracScr.Text = InfoArray(MatCount).f

TxtFracProDam.Text = InfoArray(1).b

TxtRecCon.Text = InfoArray(MatCount).rc

End Sub

Sub MnuFileExit_Click ()

    End

End Sub

```

Sub MnuFileSwipe_Click ()

Call SwipeFile

End Sub

Sub MnuInfoArb_Click ()

Call CmdOK_Click

ArbInfoFrm.show

End Sub

Sub MnuInfoMat_Click ()

Call CmdOK_Click

MatInfofrm.Show

End Sub

Sub MnuInfoPkg_Click ()

If FName <> " " Then

Close #1

FName = " "

End If

Call CmdOK_Click

ComInfoFrm.Show

MsgBox "Click ont of Type of Packge to get information"

End Sub

Sub MnuInfoPro_Click ()

If FName <> " " Then

Close #1

FName = " "

End If

Call CmdOK_Click

PPFrm.Show

End Sub

Sub OptGram_Click ()

'Assign the converter value.

ConvertPkg = 1 / 454

End Sub

Sub OptLb_Click ()

'Assign the converter value.

ConvertPkg = 1

End Sub

Swipe.Frm

Dim S1 As Integer

Dim S2 As Integer

Dim S3 As Integer

Dim S4 As Integer

Sub Timer1_Timer ()

'Display the what to do form and hide this form.

S1 = S1 + 1

If S1 > 1 Then

Timer1.Enabled = False

WhatFrm.Show

WelFrm.Hide

End If

End Sub

Sub Timer2_Timer ()

S2 = S2 + 1

If S2 > 1 Then

Timer2.Enabled = False

End If

Label1.Visible = True

End Sub

Sub Timer3_Timer ()

S3 = S3 + 1

If S3 > 1 Then

Timer3.Enabled = False

End If

Label2.Visible = True

End Sub

Sub Timer4_Timer ()

S4 = S4 + 1

If S4 > 1 Then

Timer4.Enabled = False

End If

Label3.Visible = True

End Sub

TSwipe.Frm

Sub CmdOK_Click (0

TSwipe = 0

TSwipeFrm.Hide

End Sub

Sub Form_Active ()

Call OptIn_Click

End Sub

Sub MnuFileExit_Click ()

End

End Sub

Sub MnuFileSwipe_Click ()

Call SwipeFile

End Sub

Sub MnuInfoArb_Click ()

ArbInfoFrm.Show

TSwipeFrm.Hide

End Sub

Sub InfoMat_Click ()

MatInfoFrm.Show

TSwipeFrm.Hide

End Sub

Sub MnuInfoPkg_Click ()

ComInfoFrm.Show

TSwipeFrm.Hide

MsgBox "Click one of Type of Package to get information"

End Sub

Sub MnuInfoPro_Click ()

PPFrm.Show

TSwipeFrm.Hide

End Sub

Sub MnuInfoScr_Click ()

ScrInfoFrm.Show

TSwipeFrm.Hide

End Sub

Sub MnuPrintAll_Click ()

Call Ppro

Call Pmat

Call Pswipe

Printer.Print Tab(10); "***"**

*******"**

Printer.NewPage

End Sub

Sub MnuPrintInfo_Click ()

Call Ppro

Call Pmat

Printer.Print Tab(10); "***"**

Printer.NewPage

End Sub

Sub MnuPrintResult_Click 9)

Call Ppro

Call Pswipe

Printer.Print Tab(10); "***"**

Printer.NewPage

End Sub

Sub OptCm_Click ()

'Assign the converter value.

Convert = 2.54^3

'Show calculation results.

LblTSwipe.Caption = Format(TSwipe * Convert, "Fixed")

End Sub

Sub OptIn_Click ()

'Assign the converter value.

Convert = 1

LblTSwipe.Caption = Format(TSwipe * Convert, "Fixed")

End Sub

Swipe1.Bas

Option Explicit

Type DataStruc

MatType As String * 20

Den As Single

Unit As String * 7

RRate As Single

CRate As Single

IRate Single

Ash As Single

Res As Single

W As Single

X1 As Single

X2 As Single

Y As Single

Z As Single

ProType As String * 10

Man As String * 10

ProMass As Single

ProDen As Single

PComb As Single

ProU As String * 10

Produ As String * 10

pkg As String * 10

mPkg As Single

PkgU As String *10

n As Integer

ms As Single

Scrapu As String * 10

f As Single

b As Single

rc As Single

NPro As Integer

End Type

Type MatSwipeStruc**Mat As String****VCom As Single****MScr As Single****Reuse As Single****Recy As Single****Comp As Single****Comb As Single****ProEf As Single****ScrFac As Single****PFac As Single****Swipe As Single****Convert As Single****End Type****Type InfoStruc****pkg As String****Mat As String****mPkg As Single****PkgU As String *10****n As Integer**

dPkg As String *10

Pkgdu As String *10

rr As Single

c As Single

i As Single

a As Single

e As Single

ms As Single

Scrapu As String * 10

f As Single

b As Single

rc As Single

W As Single

X1 As Single

X2 As Single

Y As Single

Z As Single

NPro As Integer

End Type

Global TMat As String

Global PComb As Single

Global dPkg As String

Global denPkg As String

Global dPr As String

Global π As Single

Global c As Single

Global i As Single

Global a As Single

Global e As Single

Global ms As Single

Global Scrapu As String

Global f As Single

Global b As Single

Global rc As Single

Global W As Single

Global X1 As Single

Global X2 As Single

Global Y As Single

Global Z As Single

Global mPkg As String

Global mPr As String

Global n As Integer

Global NPro As Integer

Global Convert As Single

Global ConvertPro As Single

Global ConvertPkg As Single

Global DataNum As Integer

Global DataRec As DataStruc

Global FName As String

Global Unit As String

Global TSwipe As Single

Global AllMat As Integer

Global MatNum As Integer

Global MatCount As Integer

Global Found As String

Global Selection As Integer

Global Mat As String

Global VCom As Single

Global MScr As Single

Global Reuse As Single

Global Recy As Single

Global Comp As Single

Global Comb As Single

Global ProEf As Single

Global ScrFac As Single

Global PFac As Single

Global Swipe As Single

Global MatSwipeArray(1 To 10) As MatSwipeStruc

Global InfoArray(1 To 10) As InfoStruc

Global num As Integer

Global ProType As String

Global PkgType As String

Global ProMass As Single

Global ProDen As Single

Global ProU As String, Produ As String

Global PkgU As String, Pkgdu As String

Global Man As Variant

Global Warn As Variant

Global Ques As Variant

Global SelectForm As Integer

Global SelectMat As String

Global AddMat As String

Global Const MB_IconExclamation = 48, MB_IconQuestion = 32

Global Const MB_IconStop = 16

Global Const MB_YesNoCancel = 3, MB_YesNo = 4

Global Const IDYes = 6, IDNo = 7, IDCcancel = 2

Sub Calculation ()

If InfoArray(MatCount).dPkg = 0 Then

Warn = MsgBox("Packaging compacted density is unable to evaluate.",

MB_IconStop, "Please Enter")

MatInfoFrm.Show

Exit Sub

End If

Dim calmPr

If MatCount = 1 Then

calmPr = mPr

Else

calmPr = 0

End If

VCom = InfoArray(MatCount).mPkg / InfoArray(MatCount).dPkg

MScr = 1 + InfoArray(MatCount).f * (InfoArray(MatCount).ms /

InfoArray(MatCount).mPkg)

Reuse = 1 + InfoArray(MatCount).W * (InfoArray(MatCount).n - 1)

Recy = (1 - InfoArray(MatCount).X1 * InfoArray(MatCount).rr) * (1 -

InfoArray(MatCount).X2 * InfoArray(MatCount).rc)

Comp = 1 - InfoArray(MatCount).Y * InfoArray(MatCount).c * (1 -

InfoArray(MatCount).e)

Comb = 1 - InfoArray(MatCount).Z * InfoArray(MatCount).i * (1 -

InfoArray(MatCount).a)


```

ProEf = 1 + InfoArray(MatCount).b

ScrFac      = InfoArray(MatCount).f * InfoArray(MatCount).ms /
`InfoArray(MatCount).dPkg * InfoArray(MatCount).rr * InfoArray(MatCount).X1

PFac = InfoArray(MatCount).b * calmPr /dPr * (1 - InfoArray(MatCount).i *
`InfoArray(MatCount).Z * PComb)

Swipe = ((VCom * MScr *Recy * Comp * Comb * ProEf / Reuse) + ScrFac +
`PFac) / NPro

End Sub

Sub Pmat ( )

Printer.Print Tab(10); "*****"
`*****"

Printer.FontBold = True

Printer.Print Tab(15); "Material Information"

Printer.Print Tab(10); "Material"; Tab(25); "Packaging"; Tab(35); "Mass";
Tab(45); "Compacted"; Tab(60); "Product Units"; Tab(77); "Number of"

Printer.Print Tab(46); "Density"; Tab(61); "per Package"; Tab(79); "Uses"

Printer.FontBold = False

MatNum = 1

Do While InfoArray(MatNum).Mat <> ""

    If Left(InfoArray(MatNum).Pkgdu, 1) = "g" Then

        Convert = 454 / (2.54^3)

```

```

Else

    Convert = 1

End If

If Left(InfoArray(MatNum).PkgU, 1) = "g" Then

    mPkg = InfoArray(MatNum).mPkg * 454

Else

    mPkg = InfoArray(MatNum).mPkg

End If

Printer.Print    Tab(12);    InfoArray(MatNum).Mat;    Tab(25);
`InfoArray(MatNum).pkg; Tab(34); mPkg; Tab(41); InfoArray(MatNum).PkgU; Tab(48);
`InfoArray(MatNum).dPkg; * Convert; Tab(52); InfoArray(MatNum).Pkgdu; Tab(64);
`InfoArray(MatNum).NPro; Tab(79); InfoArray(MatNum).n

    MatNum = MatNum + 1

Loop

Printer.Print Tab(10); "-----"
-----"

Printer.FontBold = True

Printer.Print Tab(10); "Material"; Tab(20); "Fraction"; Tab(35); "Fraction";
`Tab(50); "Fraction"; Tab(65); "Fraction"; Tab(80); "Residue"

Printer.Print Tab(20); "Recycled"; Tab(35); "Composted"; Tab(48); "Incinerated";
`Tab(67); "of Ash"; Tab(80); "Fraction"

Printer.FontBold = False

```

MatNum = 1

Do While InfoArray(MatNum).Mat < " "

Printer.Print Tab(12); InfoArray(MatNum).Mat; Tab(21); InfoArray(MatNum).rr;
 Tab(37); InfoArray(MatNum).c; Tab(52); InfoArray(MatNum).i; Tab(67);
 InfoArray(MatNum).a; Tab(82); InfoArray(MatNum).e

MatNum = MatNum + 1

Loop

Printer.Print Tab(10); "-----"
 "-----"

Printer.FontBold = True

Printer.Print Tab(10); "Material"; Tab(20); "Mass of"; Tab(35); "Fraction";
 Tab(50); "Fraction of"; Tab(65); "Recycled"

Printer.Print Tab(21); "Scrap"; Tab(35); "of Scrap"; Tab(49); "Product Damage";
 Tab(66); "Content"

Printer.Print Tab(35); "Disposed of"

Printer.FontBlod = False

MatNum = 1

Do While InfoArray(MatNum).Mat < " "

If Left(InfoArray(MatNum).Scrapu, 1) = "g" Then

ms = InfoArray(MatNum).ms * 454

End If

```

Printer.Print Tab(12); InfoArray(MatNum).Mat; Tab(21); ms; Tab(27);
`InfoArray(MatNum).Scrapu; Tab(37); InfoArray(MatNum).f; Tab(52);
`InfoArray(MatNum).b; Tab(67); InfoArray(MatNum).rc

MatNum = MatNum + 1

Loop

Printer.Print Tab(10); "-----"
`-----"

Printer.FontBold = True

Printer.Print Tab(12); "Arbitrary Information for"

Printer.Print Tab(10); "Material"; Tab(20); "Reuse"; Tab(31); "Recycling Rate";
`Tab(46); "Recycled Content"; Tab(65); "Composting"; Tab(80); "Combustion"

Printer.FontBold = False

MatNum = 1

Do While InfoArray(MatNum).Mat <> " "

Printer.Print Tab(12); InfoArray(MatNum).Mat; Tab(21);
`InfoArray(MatNum).W; Tab(37); InfoArray(MatNum).X1; Tab(52);
`InfoArray(MatNum).X2; Tab(67); InfoArray(MatNum).Y; Tab(82);
`InfoArray(MatNum).Z

MatNum = MatNum + 1

Loop

End Sub

```

Sub PPro ()

Printer.FontSize = 10

Printer.Print Tab(10); "***"**

`***"**

Printer.FontBold = True

Printer.Print Tab(10); "SWIPES values of packaging for " & ProType & " by " &

`Man

Printer.Print Tab(10); "Product Information"

Printer.Print Tab(12); "Mass " & ProMass & " " & ProU; Tab(30); "Compacted

`Density " & ProDen & " " & Produ; Tab(65); "Combustible Fraction " & Pcom

Printer.FontBold = False

End Sub

Sub PSwipe ()

Dim Unit As String

If MatSwipeArray(MatNum).Convert = 1 Then

Unit = "in^3"

Convert = 1

Else

Unit = "cm^3"

Convert = 2.54^3

End If

```

Printer.Print Tab(10); "*****"
'*****"

Printer.FontBold = True

Printer.Print Tab(15); "SWIPES values in " & Unit

Printer.Print Tab(10); "Material"; Tab(20); "Compacted"; Tab(35);
"Manufacturing"; Tab(50); "Reuse"; Tab(65); "Recycling"; Tab(80); "Composting"

Printer.Print Tab(21); "Volume"; Tab(39); "Scrap"

Printer.Bold = False

MatNum = 1

Do While MatSwipeArray(MatNum).Mat <> " "

    Printer.Print Tab(12); MatSwipeArray(MatNum).Mat; Tab(22);
'(MatSwipeArray(MatNum).VCom * Convert); Tab(39);
'MatSwipeArray(MatNum).MScr; Tab(50); MatSwipeArray(MatNum).Reuse; Tab(67);
'MatSwipeArray(MatNum).Recy; Tab(82); MatSwipeArray(MatNum).Comp

    MatNum = MatNum + 1

Loop

Printer.Print Tab(10); "-----"
'-----"

Printer.FontBold = True

Printer.Print Tab(10); "Material"; Tab(20); "Combustion"; Tab(38); "Damage";
Tab(50); "Scrap"; Tab(65); "Product"; Tab(80); "SWIPES Value"

Printer.Print Tab(34); "Compensation"; Tab(53); "Correction Factor"

```

```

Printer.FontBold = False

MatNum =1

TSwipe = 0

Do While MatSwipeArray(MatNum).Mat <> " "

    TSwipe = TSwipe + MatSwipeArray(MatNum).Swipe

    Printer.Print    Tab(12);    MatSwipeArray(MatNum).Mat;    Tab(22);
`MatSwipeArray(MatNum).Comb; Tab(38); MatSwipeArray(MatNum).ProEf; Tab(50);
`(MatSwipeArray(MatNum).ScrFac    *    Convert);    Tab(66);
`(MatSwipeArray(MatNum).PFac    *    Convert);    Tab(80);
`(MatSwipeArray(MatNum).Swipe `* Convert)

    MatNum = MatNum + 1

Loop

Printer.Print Tab(20); "Total SWIPES Value is " & TSwipe & "in^3 or "&
(TSwipe * (2.54^3)) & "cm^3"

End Sub

Sub SwipeFile ( )

    If FName <> " " Then

        Close #1

        FName = " "

    End If

    SelectForm = 4

```

```

FrmGetFile.LstFileType.Clear

FrmGetFile.LstFileType.AddItem "SWIPES Value Files (*.val)"

FrmGetFile.LstFileType.AddItem "All File (*.*)"

FrmGetFile.LstFileType.ListIndex = 0

FrmGetFile.TxtFiles.Text = "*.val"

FrmGetFile.Caption = "SWIPES Value File Selection"

FrmGetFile.Show

End Sub

Sub SwipeRec ( )

    MatNum = 1

    Do While MatNum <= 10

        If      MatSwipeArray(MatNum).Mat      =      Mat      Or

MatSwipeArray(MatNum).Mat = " " Then

            Exit Do

        End If

        MatNum = MatNum + 1

    Loop

    MatSwipeArray(MatNum).Mat = Mat

    MatSwipeArray(MatNum).VCom = VCom

    MatSwipeArray(MatNum).MScr = MScr

    MatSwipeArray(MatNum).Reuse = Reuse

```


MatSwipeArray(MatNum).Recy = Recy

MatSwipeArray(MatNum).Comp = Comp

MatSwipeArray(MatNum).Comb = Comb

MatSwipeArray(MatNum).ProEf = ProEf

MatSwipeArray(MatNum).ScrFac = ScrFac

MatSwipeArray(MatNum).PFac = PFac

MatSwipeArray(MatNum).Swipe = Swipe

End Sub

APPENDIX B

MATERIAL DATA IN STANDARD DATA FILE

APPENDIX B

MATERIAL DATA IN STANDARD DATA FILE

File name: stdmat.swp

Material¹	1	2	3	4	5	6	7
Compacted Density (Lb./in³)	.00536	.00536	.01680	.0600	.01586	.01758	.01436
Fraction Recycle	.634	.091	.555	.246	.159	.142	.019
Fraction Composted	0	0	.05	0	.05	.05	0
Fraction Incinerated	0	0	.2	0	.02	.02	.18
Fraction of Ash after Combustion	1	1	.01	1	.01	.01	.03
Fraction Residue after Compost Screen	1	1	.05	1	.01	.01	0
Reuse Value	.7	.7	.7	.8	.8	.7	.8
Recycling Rate Value	.6	.6	.6	.6	.6	.65	.45
Recycling Content Value	.5	.5	.5	.5	.5	.55	.4
Composting Value	.2	.2	.4	0	.4	.4	.2
Combustion Value	.2	.2	.3	0	.4	.4	.2

***Note:**

- 1 is Aluminum (Beer & Soft Drink Can)
- 2 is Alunimun (Others)
- 3 is Corrugated Board
- 4 is Glass
- 5 is Paper
- 6 is Paperboard
- 7 is Plastic Film

Material	8	9	10	11	12	13	14
Compacted Density (Lb./in³)	.01436	.01436	.01436	.01715	.0120	.02175	.02175
Fraction Recycle	.019	.236	.411	.047	.463	.140	0
Fraction Composted	0	0	0	0	0	.3	0
Fraction Incinerated	.18	.18	.18	.18	0	.3	.15
Fraction of Ash after Combustion	.03	.03	.03	.03	1	.3	0
Fraction Residue after Compost Screen	0	0	0	0	1	.05	0
Reuse Value	.8	.8	.8	.6	.9	.9	.7
Recycling Rate Value	.45	.45	.45	.45	.65	.4	.4
Recycling Content Value	.4	.4	.4	.4	.7	.2	.4
Composting Value	.2	.2	.2	.2	0	.3	.2
Combustion Value	.2	.2	.2	.2	0	.3	.2

*Note:

- 8 is Plastic (Rigid Containers)
- 9 is Plastic (HDPE Milk Bottles)
- 10 is Plastic (PET Soft Drink Bottles)
- 11 is Plastic (Non-film of Rigid Containers)
- 12 is Steel
- 13 is Wood
- 14 is Other Miscellaneous

MICHIGAN STATE UNIV. LIBRARIES



31293015706959