System for Processing and Analyzing WIM and AVC Data

Prepared by

Matt Folwell
Jerry Stephens

Department of Civil Engineering
Montana State University
Bozeman, Montana 59715

Prepared for

State of Montana
Department of Transportation
Research, Development, and Technology Transfer Program
in cooperation with the
U.S. Department of Transportation
Federal Highway Administration

March 1997

TECHNICAL REPORT STANDARD PAGE

1. Report No. FHWA/MT-97/8117-3	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle System for Processing and Analyzing WIM and AVC Data		5. Report Date March 1997
		6. Performing Organization Code
7. Author(s) Folwell, M., Stephens, J.E.		8. Performing Organization Report No.
9. Performing Organization Name and Address Department of Civil Engineering, Montana State University Bozeman, Montana 59717		10. Work Unit No. (TRAIS)
		11. Contract or Grant No. 8117
12. Sponsoring Agency Name and Address Montana Department of Transportation 2701 Prospect Avenue Helena, Montana 59620-1001		13. Type of Report and Period Covered Final, January 1996 to March 1997
		14. Sponsoring Agency Code 5401

15. Supplementary Notes
Research performed in cooperation with the Montana Department of Transportation and the US Department of Transportation, Federal Highway Administration.

16. Abstract

A computer program that processes Weigh-In-Motion (WIM) and Automatic Vehicle Classifier (AVC) data into a format usable by the Montana Department of Transportation (MDT) was developed in this project. The program calculates average Equivalent Single Axle Load (ESAL) factors by vehicle configuration, average vehicle weights by configuration, and indicators of WIM system performance for WIM/AVC sites. This information is available both in a selected time window and annually. The program, written using the Microsoft Access database, also identifies and describes the overweight vehicles found in the data.

17. Key Words Montana Automatic vehicle classifier, AVC Computer program Weigh-in-motion, WIM		18. Distribution Statement No Restrictions. This document is available to the public through: NTIS Springfield, Virginia 22161	
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 57	22. Price

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

Disclaimer

The opinions, findings, and conclusions expressed in this publication are those of the authors and not necessarily those of the Montana Department of Transportation or the Federal Highway Administration.

Alternate Format Statement

MDT attempts to provide reasonable accommodations for any known disability that may interfere with a person participating in any service, program, or activity of the department. Alternative accessible formats of this document will be provided upon request. For further information, call (406) 444-6269 or TTY (406) 444-7696.

ABSTRACT

A computer program that processes Weigh-In-Motion (WIM) and Automatic Vehicle Classifier (AVC) data into a format usable by the Montana Department of Transportation (MDT) was developed in this project. The program calculates average Equivalent Single Axle Load (ESAL) factors by vehicle configuration, average vehicle weights by configuration, and indicators of WIM system performance for WIM/AVC sites. This information is available both in a selected time window and annually. The program, written using the Microsoft Access database, also identifies and describes the overweight vehicles found in the data.

TABLE OF CONTENTS

Section	Page #
INTRODUCTION	1
WIM DATA COLLECTION AND ANALYSIS	2
DATA PROCESSING	6
DATA COMPARISON	24
CONCLUSION	28
RECOMMENDATIONS	28
REFERENCES	29
Appendix A - Importing WIM data into the 'WIM Data' database	A-1
Appendix B - Modification of ESAL Function	B-1
Appendix C - Modification of Weight Function	C-1

System for Processing and Analyzing WIM and AVC Data

INTRODUCTION

Background

The Montana Department of Transportation (MDT) has been collecting vehicle weight and configuration data from Weigh-In-Motion (WIM) and Automatic Vehicle Classifier (AVC) sites across the state. Four sites are presently active, with at least five additional sites scheduled for completion in the next few years. This type of data collection system provides continual information on the characteristics of the vehicles using Montana's highways, which is valuable in transportation planning, pavement design, and weight enforcement efforts. Use of WIM data eliminates the temporal considerations involved with data from static weight sites, which typically operate for only limited periods of time with respect to time of day and day of week. Additionally, static weight sites may collect little information on overweight vehicles, as such vehicles can avoid static scales by operating when these facilities are closed. A complicating feature of data from WIM and AVC sites, however, is the shear volume of data collected. The usefulness of this data can only be realized if programs are developed to process the information into a manageable format.

Objective and Scope

In this project, a computer program was developed to process WIM data to calculate, a) average vehicle weights by configuration, b) average ESAL factors by configuration, and c) WIM system performance indicators. These values are calculated for a particular WIM site for any block of data input by the user. The program is setup to accumulate data by month and year. The program also identifies overweight vehicles and their characteristics.

Program development required background research of various aspects on the WIM system performance and the raw data format. This background research included review of existing WIM systems operating within other states and evaluation of the type of system currently functioning in Montana. A system was then developed to read the raw WIM data, screen the data for anomalous entries, and calculate the desired statistics. Screening was performed through both manual observation of the raw data and review of other states screening

criteria for similar systems (1). This screening basically consisted of establishing physically credible vehicle weights (e.g. maximum/minimum values) for each vehicle configuration. Statistical processing of the WIM data was then conducted. Validation of analysis algorithms included in the program was then performed, along with some simple comparisons of the data collected by a typical WIM site with the data collected from a nearby static scale.

WIM DATA COLLECTION AND ANALYSIS

WIM Systems

There are several types of WIM systems available in the U.S., namely, piezoelectric sensor systems (system presently used in Montana), bending plate systems, capacitance pad systems, and pressure-cell systems. The most popular of these systems apparently are the piezoelectric and bending plate systems. Installation of a WIM system consists of, a) placement of the system during road construction, or b) removal of a short section/sections of pavement and placement of the WIM system in the roadway, flush with the road surface.

The piezoelectric system uses piezoelectric sensors that generate an electric current under the pressure of the axle forces of a passing vehicle. These changes in electric current are converted to axle weights based on the sensor properties. The bending plate system employs a metal plate that bends under the weight of the axles of a passing vehicle. The bending strains are measured and converted by the principles of engineering mechanics to axle weights. The bending plate system has an advantage over the piezoelectric system in that it has a larger surface area from which a signal is produced, thus limiting more of the dynamic loading effects associated with sorter measuring devices such as the piezoelectric system.

The WIM system under consideration for this project consists of two piezoelectric sensors placed about an inductance loop in each lane of traffic, with lane travel validation sensors located at the extreme sides of lane travel, as shown in Figure 1. The piezoelectric sensors collect information on vehicle weight, speed, and axle spacings. The inductance loop collects information of vehicle length. The piezoelectric sensors are installed below the pavement surface and covered with epoxy resin flush with the surface of the pavement so as to reduce the possibility of any dynamic impact of the tire on the surface of the resin generating dynamic

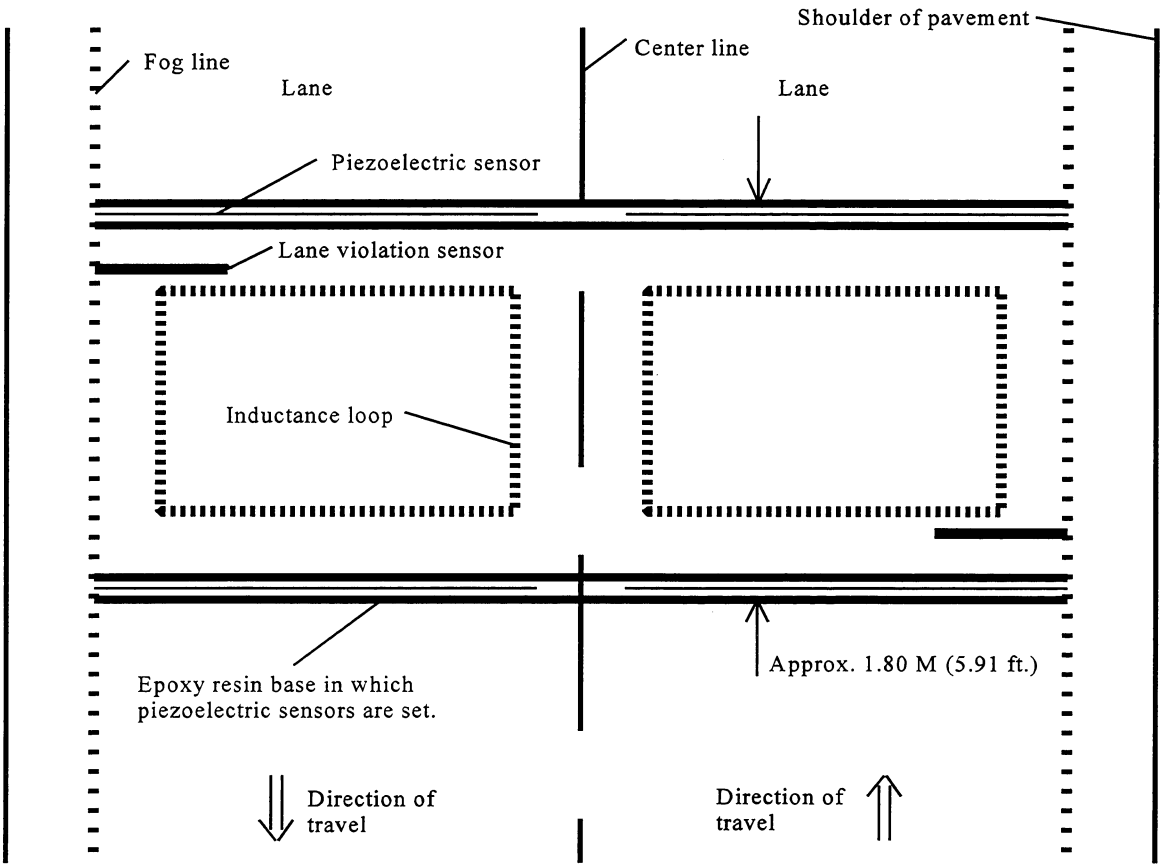


Figure 1.
Two lane setup of piezoelectric WIM system site.

loading within the sensors. The inductance loops are placed within the pavement and produce a magnetic field above the area of the sensor. The loops produce information when this magnetic field is disturbed by the passing vehicle. The validation sensor simply indicates if the vehicle traveling over the sensors is in a position such that proper data collection is possible (i.e. in a lane rather than straddling a lane). The piezoelectric sensors in combination with the inductance loop record information of individual axle weight, axle spacings, vehicle length, and vehicle speed. This information is immediately sent to an on-site computer which processes and stores information on each individual vehicle, identified as FHWA Class 4 or higher (setup of identification may be changed), that passes over the WIM site.

Information that is gathered, processed, and stored within the on-site computer for the WIM system under consideration includes: a) date and time of incident, b) lane of travel and validation/violation of that incident, c) classification and sub-classification of the particular vehicle, d) speed, e) length, f) total weight of the vehicle, and g) individual axle weights and spacings for the vehicle. All recordings are in English units of feet and pounds.

The primary discriminators of the data collected at the WIM site are the vehicle class as set by and sub-class categories as set by ECM Inc. Classification is performed according to silhouettes and tolerances input by the user (in this case, MDT). Information on individual axle weights and spacings and total vehicle length and gross weight are all used in classifying a vehicle. Vehicle class usually denotes whether the vehicle consists of a single unit, a truck and trailer, or a tractor-trailer combination with a particular number of axles (first set of characters in class designation). The sub-classification denotes different axle configurations of the particular classes recognized by the system (second set of characters in the class designation). A description of the vehicle classification presently used by MDT is presented in Table 1.

The WIM system under consideration is manufactured in France and distributed by ECM Inc. (Manor, Texas) (2). The system is designed to be self-calibrating by the on site computer, and it is expected to function fully under all types of weather conditions, with the exception of extreme cold temperatures. The self-calibration of the system occurs to correct signal drift that may occur over time. The calibration routine adjusts the system to produce the expected results for a vehicle sub-class that exhibits common characteristics of weight and length for all vehicles of the same sub-class which pass over the system. The most current systems functioning in

Table 1. ECM classification - sub-classification format.

Class-Subclass	Description of vehicle
4 - 10	2 axle single unit- passenger bus
4 - 17	3 axle single unit- passenger bus
5 - 11	2 axle single unit- 3.05 to 4.57m (10 to 15ft) distance between steer and drive axle
5 - 12	2 axle single unit- 4.57 to 6.10m (15 to 20ft) distance between steer and drive axle
5 - 18	2 axle truck and single axle full trailer- two axle truck with single axle trailer
5 - 26	2 axle truck with tandem axle full trailer- tandem axle trailer
5 - 27	2 axle truck with 2 axle full trailer- 3.05 to 4.57m (10 to 15ft) distance between steer and drive axle on truck
5 - 28	2 axle truck with 2 axle full trailer- 4.57 to 6.10m (15 to 20ft) distance between steer and drive axle on truck
5 - 34	2 axle truck with tridem axle full trailer- 3.05 to 4.57m (10 to 15ft) distance between steer and drive axle on truck
5 - 35	2 axle truck with 3 axle full trailer- 4.57 to 6.10m (15 to 20ft) distance between steer and drive axle on truck
6 - 17	3 axle single unit- truck
7 - 27	4 axle single unit- truck
7 - 36	5 axle single unit- truck
8 - 20	3 axle tractor semi- trailer- 2 axle tractor with single axle trailer
8 - 30	4 axle tractor semi- trailer- 3 axle tractor with single axle trailer
8 - 31	4 axle tractor semi- trailer- 2 axle tractor with tandem axle trailer
9 - 37	5 axle tractor semi- trailer- 3 axle tractor with tandem axle trailer
9 - 38	5 axle tractor semi- trailer- 3 axle tractor with split tandem axle trailer
9 - 39	5 axle tractor semi- trailer- 2 axle tractor with tridem axle trailer
10 - 42	6 axle tractor semi- trailer- 3 axle tractor with tridem axle trailer
10 - 45	7 axle tractor semi- trailer- 3 axle tractor with quadem axle trailer
11 - 40	5 axle tractor semi- trailer and full trailer- 2 axle tractor with single axle semi-trailer and 2 axle full trailer
12 - 43	6 axle tractor semi-trailer and full trailer- 3 axle tractor with single axle semi-trailer and 2 axle full trailer
13 - 46	7 axle tractor semi-trailer and full trailer- 3 axle tractor with single axle semi-trailer and full trailer with single lead axle and tandem axle
13 - 48	9 axle tractor semi-trailer and full trailer- 3 axle tractor with tandem axle semi-trailer and full trailer with two tandem axle sets

Montana use the steering axle of class 9 vehicles to perform this operation. Data is collected from the on site computer by means of a telecommunication link to MDT in Helena, where it is stored in the same ASCII text format that is created at the data collection site.

Uses of WIM Data

WIM systems provide information that has several uses, ranging from weight enforcement to determining pavement design parameters. WIM sites provide information on the operation of overweight trucks, which can be used to schedule enforcement activities. Information obtained from the WIM site is also used to determine pavement design parameters through data processing procedures that determine equivalent single axle load (ESAL) factors for that site. These parameters can then be used for future road construction and reconstruction that is site and state specific, alleviating the use of statewide average ESAL factors for road design.

DATA PROCESSING

Design Solution

Several different approaches to processing the raw WIM data (all with the same desired end results) were considered. These approaches were evaluated with respect to the nature of the user interface, the ease of programming, and the ability to process tens of thousands of records. Consideration was initially given to spreadsheet programs. These programs, however, were found to be limited by the amount of storage space available within the programs, themselves. Programing languages such as FORTRAN, Basic, and C++ were also considered. While these programs were able to provide all the necessary elements of interest for a solution of the problem, quality graphics were believed to be cumbersome to generate. Database programs were examined and found to provide the most efficient and attractive solution for transforming the raw WIM data into usable information.

Microsoft ACCESS(3) was the database program selected as an economical and feasibly practical program to implement. ACCESS (available with Microsoft Works) is a database program that is Windows based; therefore, most command actions are achieved by a mouse click. Creating and building a database is similar to a puzzle; there are several pieces that integrate

together to form a solution. This database solution provides a more practical tool for the user/program designer who lacks programming knowledge. This solution also shortcuts the typically lengthy code functions involved with a language-based solution.

Description of ACCESS

ACCESS is designed for a user that does not want to become involved in a complex database system. It is set up with object type tabs in a database presentation form for quick selection of desired information. These object type tabs are presented in the following order: Tables, Queries, Forms, Reports, Macros, Modules. Tables are basic spreadsheets that contain information in recordsets (in this case, records for each individual vehicle passage) which are broken down into fields. These spreadsheets may be manipulated just as any normal spreadsheet program by clicking on tool bar commands to achieve desired functions, but small mathematical commands cannot be programmed within the spreadsheet. Queries are similar to sub-tables, and generally contain information that is sorted by desired characteristics from tables of interest. Forms provide the user with both information and icons that a user may click on for execution of a desired function. This function may consist of performing a calculation, transferring data, or various other operations. Reports display information, received from a table or query, in a presentation form. Macros provide a programer/user with a quick way to execute a desired function without having to write a function to perform the same operation. Modules store all the functions that a programmer has written. These programs are written in Visual Basic, which is a quick, non-complex programming language with mass capabilities.

Once users become familiar with the ACCESS database, they will want to do more with it. Users will become interested in the availability of more information to answer yet further questions. A well designed database can handle thousands of records with little strain and quick results.

WIM ACCESS Database

In developing an ACCESS based program to process WIM data, consideration was given to the ease with which the user can adjust the existing database and expand the current

capabilities of the database. In the interest of simplifying the program function, two databases were created, one to conduct initial handling of the raw WIM data which produces information relating to the current data being processed and stored in the database, and one to handle annual WIM data information which processes and stores data progressively over the current year. Following this approach, some clutter was eliminated in the overall bulk of material within the individual databases. The basic function of the program is shown in Figures 2a and 2b, which depict the flow of the data through the elements of the database. The Weight function produces statistical information for all the truck sub-classes currently in the database. The ESAL function produces ESAL values for each individual truck record that is currently in the database, and calculates average ESAL values. The System Performance function calculates information regarding the calibration of the WIM system, which may reflect on the validity of the data provided by the system.

Data is initially obtained from the WIM system collection site in a compressed format and converted to ASCII text form by the host computer after downloading. The data is imported into the database in ASCII text form. The steps involved for the importation of external data to the 'WIM Data' database are outlined in Appendix A. To use ACCESS, the user must delete the header information that is also retrieved with the raw WIM data from the site. To accomplish this deletion, the user may wish to use a word processing program that has the capability to re-save the edited data in the same ASCII text file format. At this point, the user should make themselves aware of a current existence of a "wim data" table denoting that this table already exists and has previous WIM data information stored within it. The user is free to elect to add more data to the existing "wim data" table, or to remove the existing "wim data" table from the Table section using the Cut command on the main tool bar. This new data will then be imported to a table named "wim data" utilizing a data importation method built into the database, and importation field specifications set by the programmer. It is imperative that this table be named "wim data" as queries and procedures call information from this specific table. This step is a crucial part of the data manipulation, as the user must be familiar with the format of the data being imported and the individual fields that should appear when this step is executed. Once the importation is complete, the user is free to observe and manipulate the data now stored in the "wim data" table through tool bar commands.

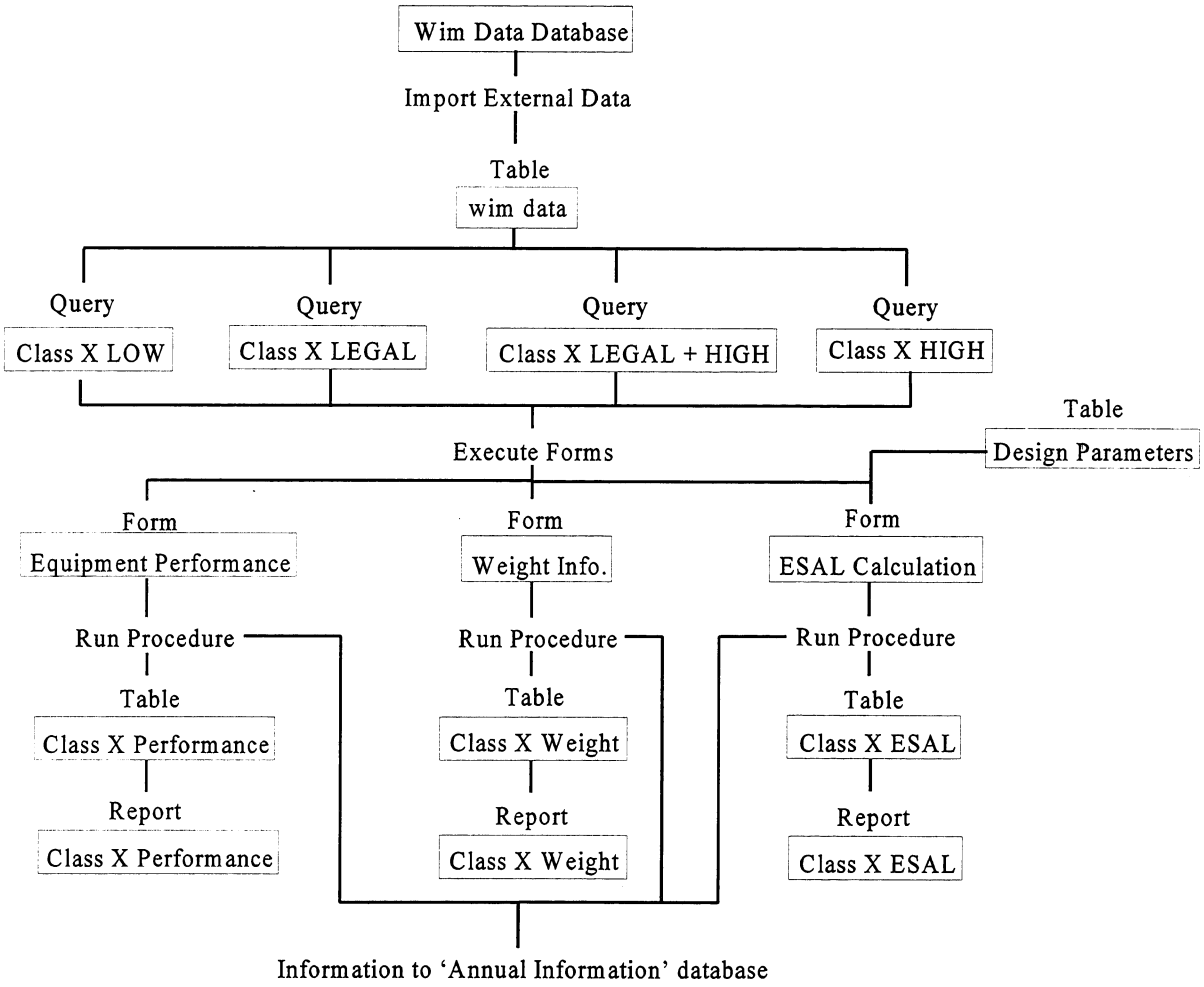


Figure 2a.
Flow of data and occurrence of events in the 'WIM Data' database.

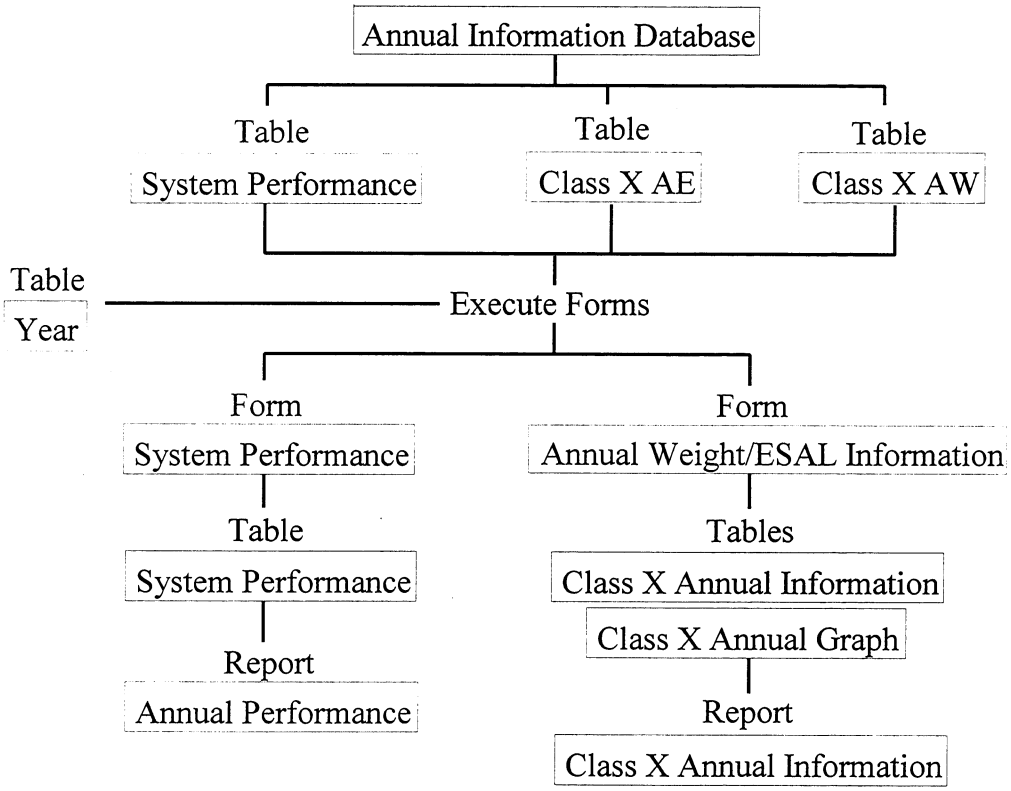


Figure 2b.
Flow of data and occurrence of events in the 'Annual Information' database.

The Forms section contains the toggles that initiate the execution of different procedures and processes. There are five forms within the 'WIM Data' database; "Weight Information", "ESAL Calculation", "System Performance", "Delete ESAL Tables", and "User Information". Within the 'Annual Information' database two forms exist; "Annual Weight / ESAL Information" and "System Performance".

The "Weight Information" form within the 'Annual Information' database provides the user with toggles that provide basic statistical weight information for a specific time period, and annually through the execution of a procedure in the Modules section. The user should be aware that information is sent to both the 'WIM Data' database, and the 'Annual Information' database upon the execution of the toggle.

The "ESAL Calculation" toggles within the 'WIM Data' database initiates procedures within the Modules section of the database. The user must click these toggles to later produce desired information. Upon clicking the toggles, the user sends desired information to tables within the 'Wim Data' database and the 'Annual Information' database. Therefore, the user should be aware that the availability of future information which is stored within the 'Annual Information database' depends upon the execution of the ESAL toggles.

The "System Performance" form within the 'Wim Data' database initiates a procedure in the Modules section that again sends information to tables in the 'WIM Data' database, and the 'Annual Information' database. The user should be aware of the significance of habitually executing this toggle over time so that a representable amount of data becomes available within the 'Annual Information' database. Clicking this toggle provides the user with an idea of how the system is currently performing by showing axle weight information of class 9-37 truck, of which typical axle weights are commonly known.

The "Delete ESAL Tables" toggles within the 'WIM Data' database initiates a delete command coded into the design of the form that removes data from ESAL storage tables established for data manipulation. A function has been written for each of these toggles that calls and executes a delete query. The function of this particular type of query is to delete recordsets contained within a table or query, without deleting the structure of the table or query. These

delete queries allow the user the option to delete previously determined ESAL information in the 'WIM Data' database stored in tables (which is used only for storage and calculational purposes in the ESAL calculation procedure) before processing a new or expanded "wim data" table. These toggles should be executed before or after a new "wim data" table has been created and before any "ESAL Calculation" toggles are executed. If these toggles are not performed, then previous ESAL information will exist within the database and execution of the "ESAL Calculation" toggles will add data to the existing information.

The "User Information" form in the 'WIM Data' database provides the user with a place to store information about activities performed with the database, and any modification made to the database. This information enables future users to retrace other users steps or actions. This information is stored continuously within the "User Information" table and is accessible either through the form or the table (and it is available for modification).

The "Annual Weight / ESAL Information" form within the 'Annual Information' database contains toggles that initiate functions which utilize the information sent to the 'Annual Information' database by the execution of ESAL and weight toggles in the 'Wim Data' database.

The "System Performance" form in the 'Annual Information' database initiates a function that calls data from tables which received data from the execution of the "System Performance" toggle in the 'Wim Data' database.

The Reports section contains the output information prepared by the toggle clicks executed within the Forms section. Three types of reports are available within the 'WIM Data' database, which represent the individual vehicle classes and system performance. These three types of reports are shown in Figures 3a, 3b, 3c. Two types of reports are available in the 'Annual Information' database which are shown in Figures 3d and 3e. All ESAL and weight statistics are available at this point for the current "wim data" table time window. The information in the ESAL reports comes from the ESAL Ave. Tables for each truck class. The information in the weight reports come from both weight tables and truck class queries. The information within the System Performance report comes from a System Performance Table. When several recordsets are present in the storage tables from where the Reports pull their information, a report will be generated for each individual recordset. The Macros section

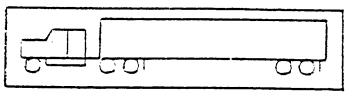
Class 9 Weight Information - Townsend Site

Time Window 1 / 4 / 96 TO 9 / 16 / 96

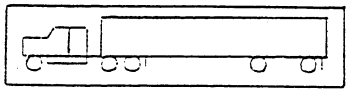
Class 9 Population: 10400 (16.11% of current wim data population)

Average Operating Weight (kips):	55.50
Average Low Weight (kips):	22.26
% Overweight:	11.38
Average Overweight (kips):	88.45

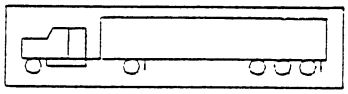
Ave. Legal Weight Sub-Class 37 (kips)	54.79
Count of legal trucks:	8240
High Weight Sub-Class 37 (kips):	117.60
Count of overweight trucks:	1120



Ave. Legal Weight Sub-Class 38 (kips)	58.89
Count of legal trucks:	1712
High Weight Sub-Class 38 (kips):	107.00
Count of overweight trucks:	64



Ave. Legal Weight Sub-Class 39 (kips)	0.00
Count of legal trucks:	0
High Weight Sub-Class 39(kips):	0.00
Count of overweight trucks:	0



Monday, January 27, 1997

Page 1 of 4

Figure 3a.
Report of Class 9 Weight Information.

Class 9 Weight Information - Townsend Site

Time Window 1 / 4 / 96 TO 9 / 16 / 96

Class 9 Population: 10400 (16.11% of current wim data population)

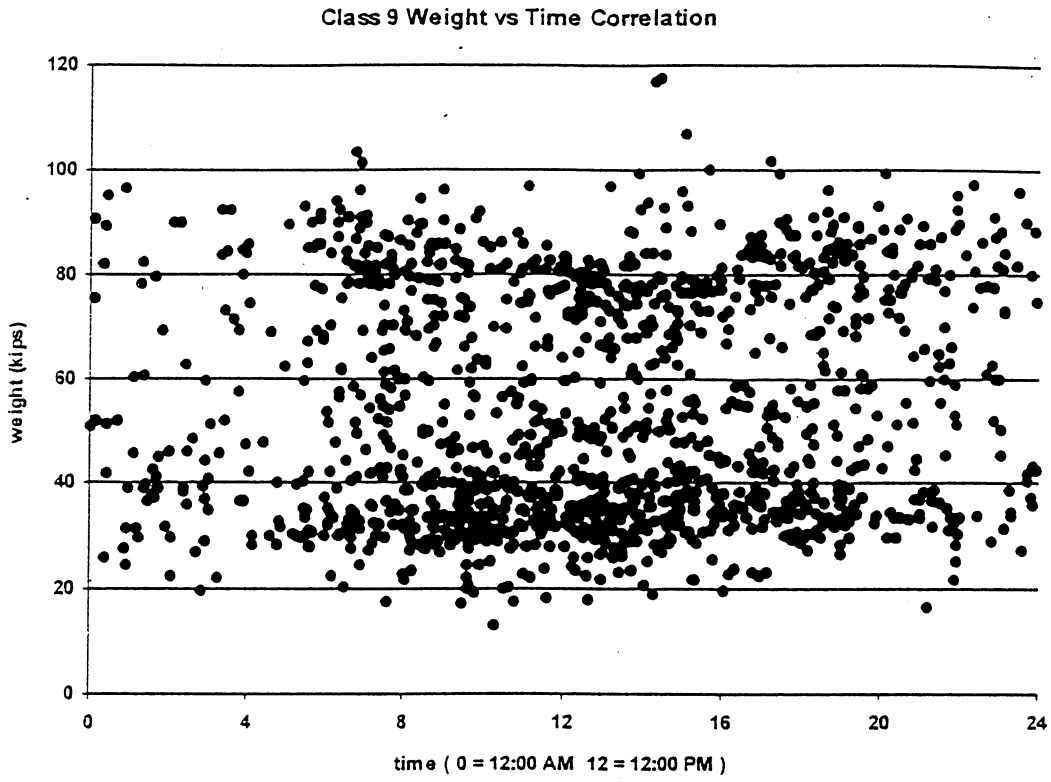


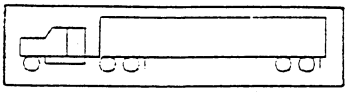
Figure 3a.
Report of Class 9 Weight Information.

Class 9 ESAL Statistical Information - Townsend Site

Time Window 1 / 4 / 96 TO 9 / 16 / 96 **Design Parameters** Pt: 2.40 SN: 3.30

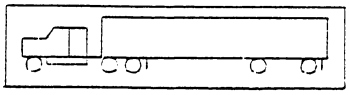
Class 9 Population: 9952 (15.42% of current wim data population)

Sub-Class 37
Population: 8240



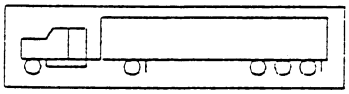
	Average	Minimum	Maximum
Steer Axle Esal:	0.06840	0.00040	0.87010
Drive Axle ESAL:	0.57390	0.00540	19.65600
Trailer Axles ESAL:	0.47100	0.00020	7.27330
Total ESAL:	1.11330	0.01630	19.66250

Sub-Class 38
Population: 1712



	Average	Minimum	Maximum
Steer Axle ESAL:	0.06100	0.00040	0.44890
Drive Axle ESAL:	0.58300	0.00650	4.60680
Lead Trailer Axle ESAL:	0.67560	0.00010	5.98490
Second Trailer Axle ESAL:	0.64890	0.00010	5.03730
Total ESAL	1.96850	0.02020	12.66530

Sub-Class 39
Population: 0



	Average	Minimum	Maximum
Steer Axle ESAL:	0.00000	0.00000	0.00000
Drive Axle ESAL:	0.00000	0.00000	0.00000
Trailer Axles ESAL:	0.00000	0.00000	0.00000
Total ESAL:	0.00000	0.00000	0.00000

Monday, January 27, 1997

Page 1 of 1

Figure 3b.
Report of Class 9 ESAL Information.

System Performance - Townsend Site

Time Window: 1 / 4 / 96 TO 9 / 16 / 96

Class 9 Population 8528

Class 9-37 Axle Weight Information

(13.21% of current wim data population)

Steer Axle Average (kips):	9.77
Steer Axle Standard Deviation (kips):	1.67
Drive Axles Average (kips):	23.71
Drive Axles Standard Deviation (kips):	9.63
Trailer Axles Average (kips):	20.23
Trailer Axles Standard Deviation (kips):	11.34
Drive + Trailer Axles Average (kips):	43.94
Drive + Trailer Axles Standard Deviation (kips):	20.52

Percentage of unclassified trucks

Percent of trucks in class 14:	26.25%
Percent of trucks in class 99:	1.65%

Monday, January 27, 1997 *Page 1 of 1*

Figure 3c.
Report of System Performance.

Class 9 Annual Information - Townsend Site

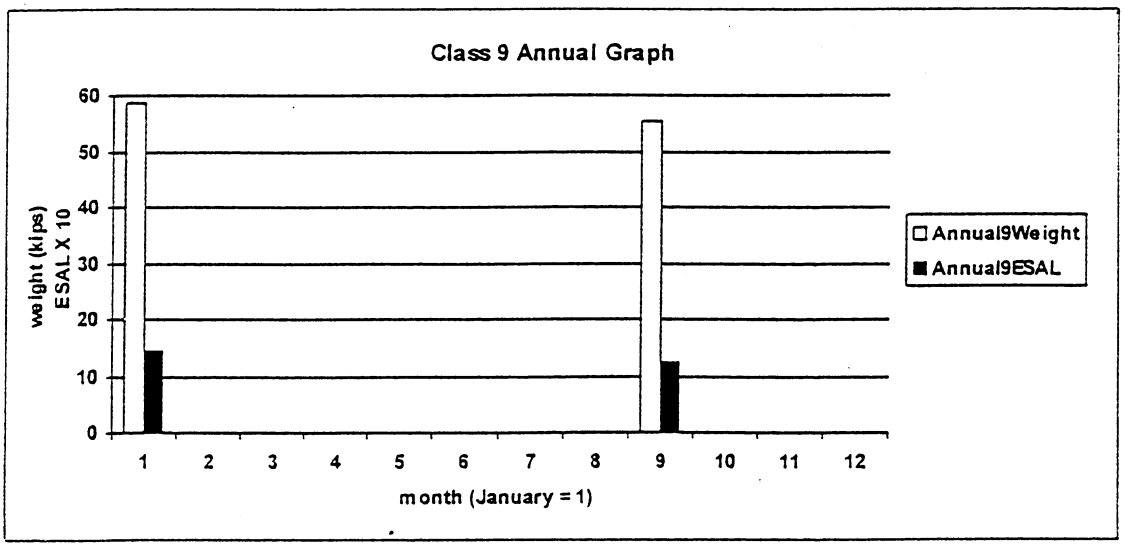
1997

	January	February	March	April	May	June
Average monthly weight	58.62500	0.00000	0.00000	0.00000	0.00000	0.00000
Trucks in weight count	64	0	0	0	0	0
Average monthly ESAL	1.44770	0.00000	0.00000	0.00000	0.00000	0.00000
Trucks in ESAL count	64	0	0	0	0	0

	July	August	September	October	November	December
Average monthly weight	0.00000	0.00000	55.48810	0.00000	0.00000	0.00000
Trucks in weight count	0	0	19840	0	0	0
Average monthly ESAL	0.00000	0.00000	1.25980	0.00000	0.00000	0.00000
Trucks in ESAL count	0	0	19840	0	0	0

(All weights in kips)

YTD weight (kips)	55.4982
YTD Trucks in Weight count	19904
YTD ESAL	1.2604
YTD Trucks in ESAL count	19904



Monday, January 27, 1997 *Page 1 of 2*

Figure 3d.
Report of Class 9 Annual Information.

Annual Performance - Townsend Site

1997

Class 9 Steer Axle Information

	January	February	March	April	May	June
Steer Axle Average	10.89	0.00	0.00	0.00	0.00	0.00
Steer Axle Std. Deviation	10.66	0.00	0.00	0.00	0.00	0.00

	July	August	September	October	November	December
Steer Axle Average	0.00	0.00	9.76	0.00	0.00	0.00
Steer Axle Std. Deviation	0.00	0.00	1.67	0.00	0.00	0.00

Percentage of Unclassified Trucks

	January	February	March	April	May	June
Class 14	19.44%	0.00%	0.00%	0.00%	0.00%	0.00%
Class 99	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%

	July	August	September	October	November	December
Class 14	0.00%	0.00%	26.27%	0.00%	0.00%	0.00%
Class 99	0.00%	0.00%	1.66%	0.00%	0.00%	0.00%

Monday, January 27, 1997 *Page 1 of 1*

Figure 3e.
Report of Annual Performance.

contains information pertaining to the report structures and form functions that initiate the opening of tables so that design and report parameters may be changed. The user does not need to interface with any of the macros.

The Modules contain the functions within procedures that are called by the toggle executions within the Forms section. These functions perform calculations such as determining ESALs for axle configurations, statistical ESAL information, statistical weight information, and system performance information. When these functions are executed within the 'WIM Data' database, they send information to both tables in the 'WIM Data' and 'Annual Information' databases for storage in tables for later access in Reports. Upon executing the Form toggles several times, several recordsets will appear in the information storage tables; current recordsets are not overwritten.

Weight Calculations

Within the individual vehicle classifications and sub-classifications, the recordsets are sorted into various weight categories using the weight field of the recordsets. Simple operations are then performed on this data to return statistical information to the truck class Weight Tables in both the 'Wim Data' database and the 'Annual Information' database. The function for this procedure is found in Appendix B. Utilizing the class and sub-class determinations made by the WIM system, itself, the ACCESS program proceeds to sort the data within each configuration category by weight categories. This breakdown first occurs into vehicle classes, and then into sub-classes. These sub-classes are then broken into weight categories of "HIGH" weight which denotes overweight trucks, a "LOW" weight category which denotes suspiciously low truck weights, a "LEGAL" weight category which includes all trucks seen as operating legally, and a "LEGAL + HIGH" weight query which is used for the calculation of ESAL values. The weight categories are shown in Table 2 and are determined as follows:

1) The "HIGH" weight category is established using the maximum legal unpermitted gross vehicle weight for each vehicle. These maximum weights are established by the bridge formula or determined by the sum of the individual permissible axle loads, of which the lower value controls. The allowable axle weights for the axle groups used in these

Table 2. Weights used in query weight divisions.

Class[1]	# of Axles	LOW truck weight[2]	HIGH truck weight[3]	Bridge Formula[4]
4 - 10	2	31.12 KN (7 kips)	177.93 KN (40 kips)	W = 500[L(2)/(2-1)+12(2)+36]
4 - 17	3	31.12 KN (7 kips)	240.20 KN (54 kips)	W = 500[L(3)/(3-1)+12(3)+36]
5 - 11	2	31.12 KN (7 kips)	177.93 KN (40 kips)	W = 500[L(2)/(2-1)+12(2)+36]
5 - 12	2	31.12 KN (7 kips)	177.93 KN (40 kips)	W = 500[L(2)/(2-1)+12(2)+36]
5 - 18	3	44.48 KN (10 kips)	240.20 KN (54 kips)	W = 500[L(3)/(3-1)+12(3)+36]
5 - 26	4	62.28 KN (14 kips)	320.27 KN (72 kips)	W = 500[L(4)/(4-1)+12(4)+36]
5 - 27	4	62.28 KN (14 kips)	355.86 KN (80 kips)	W = 500[L(4)/(4-1)+12(4)+36]
5 - 28	4	62.28 KN (14 kips)	355.86 KN (80 kips)	W = 500[L(4)/(4-1)+12(4)+36]
5 - 34	5	88.96 KN (20 kips)	366.98 KN (82.5 kips)	W = 500[L(5)/(5-1)+12(5)+36]
5 - 35	5	88.96 KN (20 kips)	366.98 KN (82.5 kips)	W = 500[L(5)/(5-1)+12(5)+36]
6 - 17	3	31.12 KN (7 kips)	240.20 KN (54 kips)	W = 500[L(3)/(3-1)+12(3)+36]
7 - 27	4	62.28 KN (14 kips)	278.01 KN (62.5 kips)	W = 500[L(4)/(4-1)+12(4)+36]
7 - 36	5	88.96 KN (20 kips)	313.60 KN (70.5 kips)	W = 500[L(5)/(5-1)+12(5)+36]
8 - 20	3	88.96 KN (20 kips)	266.89 KN (60 kips)	W = 500[L(3)/(3-1)+12(3)+36]
8 - 30	4	97.86 KN (22 kips)	329.17 KN (74 kips)	W = 500[L(4)/(4-1)+12(4)+36]
8 - 31	4	97.86 KN (22 kips)	329.17 KN (74 kips)	W = 500[L(4)/(4-1)+12(4)+36]
9 - 37	5	120.10 KN (27 kips)	391.44 KN (88 kips)	W = 500[L(5)/(5-1)+12(5)+36]
9 - 38	5	120.10 KN (27 kips)	418.13 KN (94 kips)	W = 500[L(5)/(5-1)+12(5)+36]
9 - 39	5	120.10 KN (27 kips)	366.98 KN (82.5 kips)	W = 500[L(5)/(5-1)+12(5)+36]
10 - 42	6	133.45 KN (30 kips)	429.25 KN (96.5 kips)	W = 500[L(6)/(6-1)+12(6)+36]
10 - 45	7	146.79 KN (33 kips)	464.84 KN (104.5 kips)	W = 500[L(7)/(7-1)+12(7)+36]
11 - 40	5	142.34 KN (32 kips)	444.82 KN (100 kips)	W = 500[L(5)/(5-1)+12(5)+36]
12 - 43	6	155.69 KN (35 kips)	507.10 KN (114 kips)	W = 500[L(6)/(6-1)+12(6)+36]
13 - 46	7	169.03 KN (38 kips)	569.37 KN (128 kips)	W = 500[L(7)/(7-1)+12(7)+36]
13 - 48	9	195.72 KN (44 kips)	693.92 KN (156 kips)	W = 500[L(9)/(9-1)+12(9)+36]

[1]ECM classification - sub-classification format.

[2] Weights determined by reasonable estimation and information obtained from the Florida Department of Transportation (1).

[3] Weights determined from summation of allowable loads on individual axle groups:
Single axle = 88.96 kN (20 kips)
Tandem axle set = 151.24 kN (34 kips)
Tridem axle set = 189.05 kN (42.5 kips)
Quadem axle set = 224.64 N (50.5 kips)

[4] Federal Bridge Formula[7]

$$W = 500[LN/(N-1)+12N+36]$$

where: W = (units of lbs)
L = overall length of entire vehicle.
N = number of axles of entire vehicle.

calculations are:

Single axle = 88.96 kN (20,000 lbs)
Tandem = 151.24 kN (34,000 lbs)
Tridem = 189.05 kN (42,500 lbs)
Quadem = 224.64 kN (50,500 lbs)

Bridge Formula weights were calculated as:

$$W = 500[LN/(N-1) + 12N + 36]$$

where: W = maximum weight (in lbs)
L = total length of truck (in feet)
N = # of axles of entire truck

Note that these calculations were done using only the outside-to-outside length and total number of axles.

2) The "LOW" weight category includes those vehicles that have a questionably low total weight recorded in the imported WIM data. The upper limit on this category must be determined as a reasonable minimum weight for each vehicle configuration. These limits can be estimated through reasonable estimation by the programmer. The values initially set in the program were reported in a study performed by the Florida Department of Transportation (1).
3) The "LEGAL" query category then uses the criteria of both the "LOW" and "HIGH" categories to collect the trucks that are operating legally.
4) The "LEGAL + HIGH" category includes all trucks that are operating at a weight above the minimum weight established in the "LOW" category.

The user at this point has the opportunity to adjust the lower bound weight limit established for each truck sub-classification, and modify the upper bounds, if so desired. The user also has the option to manipulate the queries and the data stored within them, just as with the tables, with tool bar options.

ESAL Calculations

Within the individual vehicle sub-classes, information is extracted from the "LEGAL + HIGH" weight category queries, and ESAL values are calculated for each vehicle. Statistical information on the ESAL values for the sample are determined. The function for this procedure is found in Appendix C. This information is sent to tables in both the 'WIM Data' database and

the 'Annual Information' database. The ESAL calculation for the individual axle loads is performed in accordance with the "AASHTO Interim Guide for Design of Pavement Structures"(4). The following Equations are coded into the functions to perform the ESAL operations:

$$G_t = \frac{\log(4.2 - p_t)}{(4.2 - 1.5)}$$

G_t = the logarithmic function of the loss in serviceability to the potential loss taken at a point where p_t (serviceability at end of time t) = 1.5 at time t, which the user can adjust in the ESAL procedures.
ρ = function of design and load variables that represents the expected number of load applications in reference to a serviceability index of 1.5.

$$\beta_x = 0.4 + \frac{0.081(L_1 + L_2)^{3.23}}{(SN+1)^{5.19}(L_2)^{3.23}}$$

$$\beta_{18} = 0.4 + \frac{0.081(18+1)^{3.23}}{(SN+1)^{5.19}(1)^{3.23}}$$

$\beta_{x,18}$ = function of design and load variables that influence the shape of the p vs. W serviceability curve.
L_1 = load on single, tandem, or tridem axle set.
L_2 = axle code (1 = single axle, 2 = tandem axle set, 3 = tridem axle set, 4 = quadem axle set).
SN = structural number design parameter.

The inverse logarithmic ESAL function for a single axle is:

$$\frac{W_{t_x}}{W_{t_{18}}} = 4.79\log(18+1) - 4.79\log(L_x+1) + \frac{G_t}{\beta_x} - \frac{G_t}{\beta_{18}}$$

W_t = axle load at end of time t.
L_x = axle load.

The inverse logarithmic ESAL function for a tandem axle set is:

$$\frac{W_{t_x}}{W_{t_{18}}} = 4.79\log(18+1) - 4.79\log(L_x+2) + 4.33(log2) + \frac{G_t}{\beta_x} - \frac{G_t}{\beta_{18}}$$

The inverse logarithmic ESAL function for a tridem axle set is:

$$\frac{W_{t_x}}{W_{t_{18}}} = 4.79\log(18+1) - 4.79\log(L_x+3) + 4.33(log3) + \frac{G_t}{\beta_x} - \frac{G_t}{\beta_{18}}$$

The inverse logarithmic ESAL function for a quadem axle set is:

$$\frac{W_{t_x}}{W_{t_{18}}} = 4.79\log(18+1) - 4.79\log(L_x+4) + 4.33(log4) + \frac{G_t}{\beta_x} - \frac{G_t}{\beta_{18}}$$

System Performance

The "System Performance" function within the 'WIM Data' database receives information from the "Class 9 - 37" query upon which the statistical weight information for the individual axle groups, such as the average group weight and standard deviation, is determined. This information is then stored in the "System Performance" table in the 'WIM Data' database; it is also sent to the "System Performance" table in the 'Annual Information' database for further processing. The "System Performance" function in the 'WIM Data' database also calls information from the "wim data" table so that a population percentage of vehicles classified as class 14 (recognizable vehicle, but not able to classify) and class 99 (unrecognizable record) can be determined. This information is stored again in both the 'WIM Data' database and the 'Annual Information' database.

The information provided within the two databases indicate current and long term trends of weight information of class 9-37 trucks, reflecting the performance of the weight recordings of the WIM system. Class 9-37 trucks were chosen for this procedure, due to the uniformity of weights that are characteristically found in class 9-37 trucks. The most common of these weights is the weight of the steer axle, which typically operate in the region of 40.03 to 53.38 kN (9,000 to 12,000 lbs). Significant deviations in actual average steer axle weights from this range may indicate a malfunction of the WIM equipment. Note, however, that system self calibration is apparently accomplished using the steer axle weights for this vehicle, which would effectively eliminate its usefulness as an independent check on system performance.

Also, the information provided within the two databases show current and long term trends of vehicle misclassification of the WIM system. A significant percentage of vehicles falling into class 14 and class 99 may indicate equipment problems and/or the existence of a new vehicle requiring that a new silhouette be added to the WIM classifier.

DATA COMPARISON

Two studies were performed on the output generated by the 'WIM Data' program:

1) the accuracy of the calculations performed by the program was verified using an independent data processing routine, and

2) the accuracy of the weight data obtained at a typical WIM site was cursorily examined with respect to the weight data obtained from static scales.

For these efforts, WIM and static weight information were collected by MDT from a region where a static scale and a WIM site are within a reasonable distance of each other. This data was collected on three different days during the months of June and September. Within these data sets, the only significant populations that existed were for class 9-37 (5 axle tractor semi-trailer) vehicles. Thus, the class 9-37 vehicles were selected to evaluate the performance of the 'WIM Data' program and to examine the performance of the WIM system, itself.

To verify the calculations done by the 'WIM Data' program, average weight and ESAL calculations were performed independently on identical data sets using the 'WIM Data' program and a Microsoft Excel spreadsheet. Calculations were done for class 9-37 vehicle data from both

the WIM and static scale sites for coincident time periods. Parameters calculated from the data include the individual axle mean weights and standard deviations, vehicle gross weights and standard deviations, and average ESALS.

The results from the spreadsheet and database calculations are compared in Table 3. All results are identical to at least 3 significant figures, which indicates that the analysis routines in the 'WIM Data' database program are functioning correctly.

Statistical comparisons of the data populations reported by the WIM and static weight systems for the same time periods were then performed to examine the accuracy of the WIM system at this site. The results of these analyses are shown in Table 4. In performing this comparison, the WIM data was screened to remove vehicles that weighed above the maximum weight found in the corresponding static weight population, as no trucks above this weight (which is substantially above the legal limit of approximately 391.6 kN (88,000 lbs)) were recorded at the static scale.

Table 3. Comparison of output values for Excel spreadsheet and 'WIM Data' database.

ESAL Information

		Steer axle	Drive axles	Trailer axles	Sum of axles
Static Data sample	WIM Database	0.19410	0.57230	0.59680	1.36320
	Excel Spreadsheet	0.19412	0.57231	0.59676	1.36319
WIM Data sample	WIM Database	0.13970	0.40670	0.37060	0.91700
	Excel Spreadsheet	0.13968	0.40671	0.37058	0.91697

Weight Information

		Steer axle	Sum of axles
Static Data sample	WIM Database	11.135	61.49
	Excel Spreadsheet	11.13478	61.48551
WIM Data sample	WIM Database	10.302	57.88
	Excel Spreadsheet	10.30222	57.88278

Table 4. Statistical information of WIM and static weight data.

Population Group A (June 7)

Static Information	Population = 69 vehicles					
Axle	Steer	Drive 1	Drive 2	Trailer 1	Trailer 2	Total
Mean Weight (kips)	11.135	12.910	12.959	12.216	12.265	61.486
Standard Deviation	0.786	3.740	3.744	4.317	4.309	16.073
WIM Information	Population = 180 vehicles					
Axle	Steer	Drive 1	Drive 2	Trailer 1	Trailer 2	Total
Mean Weight (kips)	10.302	12.099	12.407	11.220	11.854	57.833
Standard Deviation	1.446	3.054	3.243	3.662	3.770	13.805
Statistical testing						
z - test	5.808	1.608	1.079	1.697	0.697	1.667
f - test	0.859	0.846	0.888	0.816	0.900	0.864

Population Group B (June 16)

Static Information	Population = 132 vehicles					
Axle	Steer	Drive 1	Drive 2	Trailer 1	Trailer 2	Total
Mean Weight (kips)	11.427	13.387	13.436	13.051	13.105	64.406
Standard Deviation	0.618	2.972	2.968	3.891	3.891	13.357
WIM Information	Population = 316 vehicles					
Axle	Steer	Drive 1	Drive 2	Trailer 1	Trailer 2	Total
Mean Weight (kips)	10.314	11.989	12.625	11.284	12.221	58.433
Standard Deviation	1.176	2.549	2.662	2.988	3.056	11.165
Statistical testing						
z - test	13.054	4.727	2.716	4.674	2.328	4.520
f - test	0.828	0.799	0.879	0.732	0.845	0.817

Population Group C (September 11)

Static Information	Population = 102 vehicles					
Axle	Steer	Drive 1	Drive 2	Trailer 1	Trailer 2	Total
Mean Weight (kips)	11.153	12.801	12.852	12.359	12.407	61.572
Standard Deviation	0.818	3.495	3.493	4.547	4.543	16.054
WIM Information	Population = 252 vehicles					
Axle	Steer	Drive 1	Drive 2	Trailer 1	Trailer 2	Total
Mean Weight (kips)	10.185	12.283	12.255	11.400	11.932	58.057
Standard Deviation	1.251	3.195	3.184	3.656	3.784	13.982
Statistical testing						
z - test	8.596	1.294	1.493	1.896	0.933	1.934
f - test	0.847	0.917	0.900	0.820	0.892	0.880

In the results presented in Table 4, the WIM weights appear to be consistently lower than the static scale weights by approximately 10% (based on a calculated average steer axle weight of 50.15 kN (11.27 kips) for the static scale data and 45.70 kN (10.27 kips) for the WIM system). Statistical comparison tests were run on the WIM and static weight population samples, to determine if there is a difference between the two populations. These tests were performed assuming two independent samples with unequal standard deviations and a 5% significance level. First, a z-test was performed on the sample means. The assumption of identical means must be rejected if the z statistic exceeds a value of 1.96, which is the case for all steer axle comparisons in Table 4. Thus, a statistically significant difference between the means of the WIM and static steer axle weight populations exists. Secondly, a f-test was performed on the variances of the two sample populations to determine if the standard deviations were similar. All of the values calculated fall below the corresponding critical f-test value at a 5% significance level. Thus, the standard deviations of the WIM and static weight data populations are statistically similar.

The apparent reason for the difference between the axle weights reported by the WIM and the static scale is the use of an incorrect calibration factor in the WIM system. The WIM system being used, as described earlier, self-calibrates using the steer axle of a class 9-37 vehicle. This self-calibration is intended to correct any signal drift by the system. To initially calibrate the installed system, it is recommended in the installation manual for the WIM system that static weights of steer axles from the intended calibration vehicle be obtained at the installation site from a population of 100+ vehicles (2). The mean steer axle weight and standard deviation is then computed from this static weight population, and input into the WIM system. The system will then automatically calibrate itself on a continual basis using the user input mean weight value. The static scale data would suggest that a steer axle weight of approximately 50.2 kN (11.3 kips) may be appropriate at this site, although further investigation would be necessary to establish a new calibration value. Presuming the WIM system is functioning properly, this calibration value now appears to be set at approximately 47.2 kN (10.3 kips), consistent with the value preset by the manufacturer.

CONCLUSIONS

WIM data will provide MDT with valuable information on vehicle weight statistics and vehicle pavement demands at locations around the state. WIM sites collect data on a continual basis, thus providing information to MDT that is not normally attainable with the existing data collection system, namely, static weight stations.

A computer program was developed to process this WIM data into usable information for the MDT. This computer program was built in a database program called Microsoft ACCESS, which has the capabilities to handle tens of thousands of data records. This type of program was chosen due to its ease of use to both a user and a programmer, and its excellent graphical capabilities. ACCESS is a Windows based program, so most commands given by the user are achieved by a mouse click. The program processes WIM data to obtain average weight and ESAL information for the traffic stream at the WIM site.

RECOMMENDATIONS

The information desired from by the WIM database system will continue to evolve as users of the information gain a feel for the different types of information that the WIM system can provide. Information of interest not delivered under the present system, for example, includes information of Annual Average Daily Traffic (AADT), where reports would be produced to show traffic counts on a monthly and annual basis. Routines could also be developed to directly support weight enforcement efforts. For example, reports showing the correlation between weight enforcement activities and observed volumes of overweight vehicle traffic can be made. It may also be useful to compare static weight station data and WIM data to further validate WIM system performance and possibly develop correlation factors for use with static weight station data in areas where WIM data is unavailable.

REFERENCES

1) Reel, Richard L. "Automated Editing of Traffic Data in Florida" National Traffic Data Acquisition Conference, Proceedings Volume 2, pp 143-164, September 18-22, 1994.

2) White, R. And Smith, S., ECM Inc., 10400 Block, Hwy 290E, P.O. Box 888, Manor, Texas, 78653.

3) Browne, A. and Balter, A., "Essential ACCESS 95" first edition, SAMS Publishing, 1995.

4) "AASHTO Interim Guide for Design of Pavement Structures 1972" Chapter 3 revised, 1981, American Association of State Highway and Transportation Officials, pp 59 - 63, 1981.

Appendix A

Importing WIM Data into the 'WIM Data' Database

Method and Modification for Importation of Raw Data into ACCESS Database

1.) Begin by going to **file** in the main menu of Access and select **Get External Data**, then select **Import** in the sub-menu that appears.

2.) A file import form now appears so that a file from a folder may be selected for importation. There are two file selection boxes in the lower left hand corner of the import form in which a file name can be typed and file format type must be designated so that the data enters Access in the proper format. Once a file has been selected either by highlighting the file or typing the file name and **Text Files** has been selected in the file type, the **Import** icon can be clicked.

3.) A Text Import Wizard form now appears with a sample of the data that is being imported into the database and format type. At this point the **Advanced...** icon is to be selected so that field designations for the data being imported can be made. An Import Specification form now appears with the current field specifications for the table to which data will be imported. The **Specs...** now needs to be clicked upon which a window will open containing the names of field specifications set by the programmer. The "Import Specifications" file needs to be opened, this file contains the following fields;
month, day, year, hour, minute, second, vhnum, a2, a1, , a3, sn, ln, val, viol, ca, cca, speed, length, esal, todt, twt2, twt1, twt3, dtb1, wt12, wt11, wt13, dt12, wt 22, wt21, wt23, dt23, wt32, wt31, wt33, dt34, wt42, wt41,wt43, dt45, wt52, wt51, wt53, dt56, wt62, wt61, wt63, dt67 wt72, wt71, wt73, dt78, wt82, wt81, wt83, dt89, wt92, wt91, wt93.
ca = truck class
cca = truck sub-class
Next click on **O.K.** icon to return to the Text Import Wizard.

4.) Now click on the **Next>** icon which will then show the field breaks and titles in the strings of data being imported.

5.) Click **Next>** again, the Text Import Wizard form now ask's where the data is to be stored and gives two options of storing the data in an existing table or storing the data in a new table. If the user wants to add more WIM data to information already existing in the "wim data" table, then the user needs to type "wim data" in the existing table selection box. Now click the **Next>** icon which will either send the user to the next step or to the final step if the data is being imported into an already existent "wim data" table.

6.) The Import form now displays field information modification that can be made at this point. The user should not need to modify any of the import field designations here. Now click the **Next>** icon.

7.) The next form ask's the user if a primary key should be added to the data that is being imported. A primary key is a numbering of all the recordsets of the data being imported in

ascending order. There is no need for this, so at this point the user should select No Primary Key and click on the **Next>** icon.

8.) The current form now asks for the name of the new table into which the data being imported will be placed. The user should type "wim data" in the Import to Table box, and click on the finish icon. There are also two import option selection boxes that appear in this form, one that analyzes the data being imported, and another that brings up a help box once the data has been imported. There should be no need for the user to initiate either of these two options.

Appendix B

Modification of ESAL Function

Procedure for modification of ESAL function within the 'WIM Data' database for adding new truck classes and sub-classes.

***X* denotes truck class**
***xx* denotes truck sub-class**

Italics denotes user assigned number (class or sub-class) or that a user specified character should be substituted.

1.) For producing an entire new function for a new vehicle class that is not already coded, a new function must be declared by typing the following within a new procedure;

Function ESALClass*X*()

Or if a new sub-class is desired to be added, continue to the next step.

2.) Queries and/or tables must now be opened so that the function is able to call and send information with these. This operation is accomplished with the following set of commands in which a query containing the desired weight information is opened, and several tables are opened to send information requiring storage for later use in the function. One of these tables stores information that is used by the Class *X* ESAL report, and the other tables are used to store information needed later in the function. The following is an example for a single vehicle sub-class, multiple vehicle sub-classes follow in the same manner.

```
Dim dbxx As DATABASE
Dim rsxx As Recordset
Dim dbesalxx As DATABASE
Dim rsesalxx As Recordset
Dim dbave As DATABASE
Dim rsave As Recordset

Set dbxx = CurrentDb()
Set rsxx = dbxx.OpenRecordset("Class X - xx LEGAL + HIGH")
Set dbesalxx = CurrentDb()
Set rsesalxx = dbesalxx.OpenRecordset("Class X-xx ESAL")
Set dbave = CurrentDb()
Set rsave = dbave.OpenRecordset("Class X ESAL Ave")
```

The following step opens the design parameters table so that the function can access the pavement design parameters of p_t and SN set in the "ESAL Calculation" form.

```
Dim dbdp As DATABASE
Dim rsdp As Recordset
```

```
Set dbdp = CurrentDb()
Set rsdp = dbdp.OpenRecordset("Design Parameters")
```

3.) Variables used in the ESAL calculation must now be dimensioned to accommodate the type of information they will hold. The following is a dimensioning of variables for a single vehicle sub-class, multiple vehicle sub-classes follow similarly. The programmer should use caution in naming sub-class and counter variables at this point so as not to use identical variable names between vehicle sub-classes.

```
Dim Gt, B1, B2, b, Bx11, Bx21, Bx31, Bx12, Bx22, Bx32, Bx1, Bx2, Bx3 As Currency
Dim SE, SEE, SE1, SE2, SE3, SE4, SE5, DE, DEE, DE1, DE2, DE3, DE4, DE5, TE,
    TEE, TE1, TE2, TE3, TE4, TE5, TOTALTxx As Currency
Dim Sxx, Dxx, Txx, TOTALxx As Currency
Dim TOTALxxJan, TOTALxxFeb, TOTALxxMar, TOTALxxApr, TOTALxxMay,
    TOTALxxJun, TOTALxxJul, TOTALxxAug, TOTALxxSep, TOTALxxOct,
    TOTALxxNov, TOTALxxDec As Currency
Dim z, zJan, zFeb, zMar, zApr, zMay, zJun, zJul, zAug, zSep, zOct, zNov, zDec
    As Integer
```

4.) Next the variables must initially be set to zero so that any previously existing information that these variables have been set to is eliminated. The following is an example for a 5 axle tractor-trailer unit with a steer axle, drive tandem, and trailer tandem.

```
Let z = 0
Let zJan = 0
Let zFeb = 0
Let zMar = 0
Let zApr = 0
Let zMay = 0
Let zJun = 0
Let zJul = 0
Let zAug = 0
Let zSep = 0
Let zOct = 0
Let zNov = 0
Let zDec = 0
Let Sxx = 0
Let Dxx = 0
Let Txx = 0
Let TOTALxx = 0
Let TOTALTxx = 0
Let TOTALxxJan = 0
Let TOTALxxFeb = 0
```

Let TOTALxxMar = 0
Let TOTALxxApr = 0
Let TOTALxxMay = 0
Let TOTALxxJun = 0
Let TOTALxxJul = 0
Let TOTALxxAug = 0
Let TOTALxxSep = 0
Let TOTALxxOct = 0
Let TOTALxxNov = 0
Let TOTALxxDec = 0

5.) ESAL equation parameters of Gt and β can now be calculated and set. These two variables remain constant for all truck classes and sub-classes, therefore this calculation only needs to be performed once.

Gt = (Log((4.2 - (rsdp![Pt])) / (4.2 - 1.5)) / (Log(10#)))
B1 = 0.081 * ((18 + 1) ^ 3.23)
B2 = ((rsdp![SN] + 1) ^ 5.19) * (1 ^ 3.23)
b = 0.4 + (B1 / B2)
Gt = loss of serviceability to potential loss.
B1 = Numerator of beta equation
B2 = Denominator of beta equation
b = (beta) Design influence of p versus W curve

6.) Next the ESAL calculation for the individual axle weights within the query of recordsets containing the weight information is performed. To accomplish this, the recordsets within the query needs to be looped upon, required information calculated for each axle group, and then stored for later use within the function or report.

6a.) The first part of the loop code calculates the β_x parameter for the individual axle groups which changes with the different axle weights that are used in the equations. The following example is for the same five axle setup as previously described.

Bx for steer axle
Bx11 = 0.081 * ((rsxx![wt1] + 1) ^ 3.23)
Bx12 = ((rsdp![SN] + 1) ^ 5.19) * (1 ^ 3.23)
Bx1 = 0.4 + (Bx11 / Bx12)
Bx for tandems
Bx21 = 0.081 * ((rsxx![wt2] + rsxx![wt3] + 2) ^ 3.23)
Bx22 = ((rsdp![SN] + 1) ^ 5.19) * (2 ^ 3.23)
Bx2 = 0.4 + (Bx21 / Bx22)
Bx31 = 0.081 * ((rsxx![wt4] + rsxx![wt5] + 2) ^ 3.23)
Bx32 = ((rsdp![SN] + 1) ^ 5.19) * (2 ^ 3.23)

Bx3 = 0.4 + (Bx31 / Bx32)

6b.) The next step is code that calculates the ESAL's for the individual axle groupings. This ESAL calculation is performed by breaking the inverse logrithmic function into five parts, followed by a summation of these five pieces. Also, a running total is performed For use later in the function.

ESAL EQN. for steer axle

SE1 = 4.79 * (Log(19) / Log(10#))
SE2 = 4.79 * ((Log(rs*xx*![wt1] + 1)) / (Log(10#)))
SE3 = 0
SE4 = Gt / Bx1
SE5 = Gt / b
SE = SE1 - SE2 + SE4 - SE5
SEE = 1 / (10 ^ (SE))
S*xx* = S*xx* + SEE

ESAL for tandem sets

DE1 = 4.79 * (Log(19) / Log(10#))
DE2 = 4.79 * ((Log(rs*xx*![wt2] + rs*xx*![wt3] + 2)) / (Log(10#)))
DE3 = 4.33 * (Log(2) / Log(10#))
DE4 = Gt / Bx2
DE5 = Gt / b
DE = DE1 - DE2 + DE3 + DE4 - DE5
DEE = 1 / (10 ^ (DE))
D*xx* = D*xx* + DEE
TE1 = 4.79 * (Log(19) / Log(10#))
TE2 = 4.79 * ((Log(rs*xx*![wt4] + rs*xx*![wt5] + 2)) / (Log(10#)))
TE3 = 4.33 * (Log(2) / Log(10#))
TE4 = Gt / Bx3
TE5 = Gt / b
TE = TE1 - TE2 + TE3 + TE4 - TE5
TEE = 1 / (10 ^ (TE))
T*xx* = T*xx* + TEE
TOTAL*xx* = SEE + DEE + TEE

6c.) The next code segment prepares data that is later sent to the 'Annual Information' database. This segment of code breaks information into individual months.

```
If rsxx![month] = 1 Then
  zJan = zJan + 1
  TOTALxxJan = TOTALxxJan + TOTALxx
ElseIf rsxx![month] = 2 Then
  zFeb = zFeb + 1
```

```
  TOTALxxFeb = TOTALxxFeb + TOTALxx
ElseIf rsxx![month] = 3 Then
  zMar = zMar + 1
  TOTALxxMar = TOTALxxMar + TOTALxx
ElseIf rsxx![month] = 4 Then
  zApr = zApr + 1
  TOTALxxApr = TOTALxxApr + TOTALxx
ElseIf rsxx![month] = 5 Then
  zMay = zMay + 1
  TOTALxxMay = TOTALxxMay + TOTALxx
ElseIf rsxx![month] = 6 Then
  zJun = zJun + 1
  TOTALxxJun = TOTALxxJun + TOTALxx
 ElseIf rsxx![month] = 7 Then
  zJul = zJul + 1
  TOTALxxJul = TOTALxxJul + TOTALxx
ElseIf rsxx![month] = 8 Then
  zAug = zAug + 1
  TOTALxxAug = TOTALxxAug + TOTALxx
ElseIf rsxx![month] = 9 Then
  zSep = zSep + 1
  TOTALxxSep = TOTALxxSep + TOTALxx
ElseIf rsxx![month] = 10 Then
  zOct = zOct + 1
  TOTALxxOct = TOTALxxOct + TOTALxx
ElseIf rsxx![month] = 11 Then
  zNov = zNov + 1
  TOTALxxNov = TOTALxxNov + TOTALxx
ElseIf rsxx![month] = 12 Then
  zDec = zDec + 1
  TOTALxxDec = TOTALxxDec + TOTALxx
End If
  TOTALTxx = TOTALTxx + TOTALxx
```

6d.) The next section of code sends individual axle group ESAL information for each recordset that the loop iterates upon to the truck sub-class ESAL tables.

```
With rsesalxx
.AddNew
![SA xx ESAL] = SEE
![DA xx ESAL] = DEE
![TA xx ESAL] = TEE
![TOTAL ESAL xx] = TOTALxx
```

```
.UPDATE
End With
```

6e.) The loop execution begins and ends with the following commands. x represents a counter that is later used in the calculation of certain information.

Placed before step 6a.

```
Do While Not rsxx.EOF
x = x + 1
```

Placed after step 6e.

```
rsxx.MoveNext
Loop
rsxx.Close
```

7.) The following code sends information to the "Class *X* ESAL Ave." table using information stored in variables by the segments of code previously described. This segment of code also calls information from the ESAL tables established earlier, and from the initial "wim data" table. The code is setup with only one truck sub-class presented. For additional sub-classes, segments of code lacking the "aa" addition need to be copied and parameters changed.

```
With rsave
.AddNew
If z > 0 Then
![AveSA xx] = Sxx / z
![AveDA xx] = Dxx / z
![AveTA xx] = Txx / z
![AveTOTAL xx] = TOTALTxx / z
Else
![AveSA xx] = 0
![AveDA xx] = 0
![AveTA xx] = 0
![AveTOTAL xx] = 0
End If
If z > 0 Then
![sminxx] = DMin("[SA xx ESAL]", "Class X-xx ESAL")
![smaxxx] = DMax("[SA xx ESAL]", "Class X-xx ESAL")
![dminxx] = DMin("[DA xx ESAL]", "Class X-xx ESAL")
![dmaxxx] = DMax("[DA xx ESAL]", "Class X-xx ESAL")
![tminxx] = DMin("[TA xx ESAL]", "Class X-xx ESAL")
![tmaxxx] = DMax("[TA xx ESAL]", "Class X-xx ESAL")
![ttminxx] = DMin("[TOTAL ESAL xx]", "Class X-xx ESAL")
```

```
![ttmaxxx] = DMax("[TOTAL ESAL xx]", "Class X-xx ESAL")
Else
![sminxx] = 0
![smaxxx] = 0
![dminxx] = 0
![dmaxxx] = 0
![tminxx] = 0
![tmaxxx] = 0
![ttminxx] = 0
![ttmaxxx] = 0
End If
![Count] = z + (aa)      {(aa) if addition truck sub-classes exist}
![Perc] = ((z + (aa)) / (DCount("[day]", "wim data")))
![d1] = DFirst("[day]", "wim data")
![d2] = DLast("[day]", "wim data")
![m1] = DFirst("[month]", "wim data")
![m2] = DLast("[month]", "wim data")
![y1] = DFirst("[year]", "wim data")
![y2] = DLast("[year]", "wim data")
![Pt] = rsdp![Pt]
![SN] = rsdp![SN]
![countxx] = z
![countyy] = (aa)
.UPDATE
End With
```

8.) The next segment of code sends information to the 'Annual Information' database. As initially done, tables need to be opened for data storage, and variables need to be dimensioned.

```
Dim dbanX As DATABASE
Dim rsanX As Recordset
Set dbanX = OpenDatabase("c:\my documents\Annual Info.mdb")
Set rsanX = dbanX.OpenRecordset("Class X AE")

Dim AExxJ, AExxF, AExxMR, AExxAP, AExxMY, AExxJN, AExxJL, AExxAU,
      AExxS, AExxO, AExxN, AExxD As Currency
Dim CountxxJ, CountxxF, CountxxMR, CountxxAP, CountxxMY, CountxxJN,
      CountxxJL, CountxxAU, CountxxS, CountxxO, CountxxN, CountxxD
      As Currency
```

9.) Next, values are calculated and sent to a table in the 'Annual Information' database for later usage in a function located in the 'Annual Information' database. The example below

contains code for sending information of both single and multiple truck sub-classes using a weighted average method to determine values.

```
With rsanX
.AddNew
```

Process copied for each month of the year using variables dimensioned above.
For a single truck sub-class;

```
If zJan > 0 Then
![CaXAEJan] = TOTALxxJan / zJan
![CountXJan] = zJan
Else
![CaXAEJan] = 0
![CountXJan] = 0
End If
```

For multiple truck sub-classes, dimension further variables.

```
Dim AWaaJ ..... As Currency
Dim CountaaJ ..... As Currency

If zJan > 0 Then
AExxJ = TOTALxxJan / zJan
CountxxJ = zJan
End If
If yJan > 0 Then
AEaaJ = TOTALaaJan / yJan
CountaaJ = yJan
End If
If (CountxxJ + CountaaJ) > 0 Then
![CaXAEJan] = ((AExxJ * CountxxJ) + (AEaaJ * CountaaJ)) / (CountxxJ + CountaaJ)
![CountXJan] = CountxxJ + CountaaJ
End If

.UPDATE
End With
```

10.) The final step of the ESAL code is to close all of the open tables and queries that have not been closed to this point. This is accomplished by the following commands.

```
rsesalxx.Close
rsesalaa.Close
rsave.Close
rsanX.Close
```

Appendix C

Modification of Weight Function

Procedure for modification of Weight function within the 'WIM Data' database for adding new truck classes and sub-classes.

***X* denotes truck class**
***xx* denotes truck sub-class**

Italics denotes user assigned number (class or sub-class) or that a user specified character should be substituted.

1.) For producing a entire new function for a new truck class that is not already coded, a new function must be declared by typing the following within a new procedure;

Function WeightClass*X*()

Or if a new sub-class is desired to be added, continue to the next step.

2.) Queries and/or tables must now be opened so that the function is able to call and send information with these. This step is accomplished with the following set of commands in which a query containing the desired weight information is opened, and a table is opened for storage to which information will be sent for later use. This table stores information that is used by the "Class *X* Weight" report. The following is an example for a single truck sub-class. Dimensioning of the different tables used throughout the function is performed first, the commands for opening the different tables will be presented at the points in the code where information is needed from the tables or queries. Multiple truck sub-classes follow in the same manner.

Dim dbclass*Xxx* As DATABASE
Dim rsclass*Xxx* As Recordset
Dim dblow*xx* As DATABASE
Dim rslow*xx* As Recordset
Dim dblegalhigh*xx* As DATABASE
Dim rslegalhigh*xx* As Recordset
Dim dbhigh*xx* As DATABASE
Dim rshigh*xx* As Recordset

Dim dbstat As DATABASE
Dim rsstat As Recordset

Opening record for calculated statistic storage.
Set dbstat = CurrentDb()
Set rsstat = dbstat.OpenRecordset("Class *X* Weight")

3.) The following section of code opens the "LEGAL + HIGH" query to receive data.

Two variables are declared and set to zero, which are used as a counter and running sum. The variable *z* is used as a loop counter for the truck sub-class. AVEOW*xx* is an average operating weight summation for the truck sub-class. AVEOW is a variable for the storage of the singular or weighted summation of the truck sub-classes. This step is iterated for multiple truck sub-classes.

Calculating average operating weight of all trucks accepted above a minimum weight cutoff.

Set dblegalhigh*xx* = CurrentDb()
Set rslegalhigh*xx* = dblegalhigh*xx*.OpenRecordset("Class *X* - *xx* LEGAL + HIGH")

Dim *z* As Integer
Dim AVEOW As Currency
Dim AVEOW*xx* As Currency

Let *z* = 0
Let AVEOW*xx* = 0

Annual operating weight information for sub-class *xx*.

Dim TOTAL*xx*Jan, TOTAL*xx*Feb, TOTAL*xx*Mar, TOTAL*xx*Apr, TOTAL*xx*May, TOTAL*xx*Jun, TOTAL*xx*Jul, TOTAL*xx*Aug, TOTAL*xx*Sep, TOTAL*xx*Oct, TOTAL*xx*Nov,TOTAL*xx*Dec As Currency
Dim *z*Jan, *z*Feb, *z*Mar, *z*Apr, *z*May, *z*Jun, *z*Jul, *z*Aug, *z*Sep, *z*Oct, *z*Nov, *z*Dec As Integer
Let *z*Jan = 0
Let *z*Feb = 0
Let *z*Mar = 0
Let *z*Apr = 0
Let *z*May = 0
Let *z*Jun = 0
Let *z*Jul = 0
Let *z*Aug = 0
Let *z*Sep = 0
Let *z*Oct = 0
Let *z*Nov = 0
Let *z*Dec = 0
Let TOTAL*xx*Jan = 0
Let TOTAL*xx*Feb = 0
Let TOTAL*xx*Mar = 0
Let TOTAL*xx*Apr = 0
Let TOTAL*xx*May = 0
Let TOTAL*xx*Jun = 0
Let TOTAL*xx*Jul = 0
Let TOTAL*xx*Aug = 0
Let TOTAL*xx*Sep = 0

```
Let TOTALxxOct = 0
Let TOTALxxNov = 0
Let TOTALxxDec = 0
```

3a.) The following segment of code loops on the recordsets of the open query to prepare information to be sent to the 'Annual Information' database. This code is iterated for multiple truck sub-classes. The loop is initiated by a DO command with the proper loop proceedings.

```
Do While Not rslegalhighxx.EOF
z = z + 1
AVEOWxx = AVEOWxx + rslegalhighxx![twt]

If rslegalhighxx![month] = 1 Then
   zJan = zJan + 1
   TOTALxxJan = TOTALxxJan + rslegalhighxx![twt]
ElseIf rslegalhighxx![month] = 2 Then
  zFeb = zFeb + 1
  TOTALxxFeb = TOTALxxFeb + rslegalhighxx![twt]
ElseIf rslegalhighxx![month] = 3 Then
  zMar = zMar + 1
  TOTALxxMar = TOTALxxMar + rslegalhighxx![twt]
ElseIf rslegalhighxx![month] = 4 Then
  zApr = zApr + 1
  TOTALxxApr = TOTALxxApr + rslegalhighxx![twt]
ElseIf rslegalhighxx![month] = 5 Then
  zMay = zMay + 1
  TOTALxxMay = TOTALxxMay + rslegalhighxx![twt]
ElseIf rslegalhighxx![month] = 6 Then
  zJun = zJun + 1
  TOTALxxJun = TOTALxxJun + rslegalhighxx![twt]
ElseIf rslegalhighxx![month] = 7 Then
  zJul = zJul + 1
  TOTALxxJul = TOTALxxJul + rslegalhighxx![twt]
ElseIf rslegalhighxx![month] = 8 Then
  zAug = zAug + 1
  TOTALxxAug = TOTALxxAug + rslegalhighxx![twt]
ElseIf rslegalhighxx![month] = 9 Then
  zSep = zSep + 1
  TOTALxxSep = TOTALxxSep + rslegalhighxx![twt]
ElseIf rslegalhighxx![month] = 10 Then
  zOct = zOct + 1
  TOTALxxOct = TOTALxxOct + rslegalhighxx![twt]
```

```
ElseIf rslegalhighxx![month] = 11 Then
 zNov = zNov + 1
 TOTALxxNov = TOTALxxNov + rslegalhighxx![twt]
ElseIf rslegalhighxx![month] = 12 Then
 zDec = zDec + 1
 TOTALxxDec = TOTALxxDec + rslegalhighxx![twt]
End If
rslegalhighxx.MoveNext
Loop
```

4.) The next segment of code places information in the AVEOW variable. Code is shown for both a single truck sub-class and for multiple truck sub-classes.

Performing weight calculation for a single truck sub-class.

```
If x > 0 Then
AVEOW = AVEOWxx / z
Else
AVEOW = 0
End If
```

Perform weighted average on weight for several truck sub-classes.
AVEOW*aa* is calculated the same way as AVEOW*xx*, but new truck sub-class.
Count for new sub-class = *y*.

```
Dim AVEOWx, AVEOWy As Currency
If x > 0 Then
AVEOWx = AVEOWxx / z
Else
AVEOWx = 0
End If
If y > 0 Then
AVEOWy = AVEOWaa / y
Else
AVEOWy = 0
End If
If (z + y) > 0 Then
AVEOW = ((AVEOWx * z) + (AVEOWy * y)) / (z + y)
Else
AVEOW = 0
End If
```

5.) The next code segment opens a query called "Class *X* - *xx* LOW" to obtain information to perform the same type of operation described in the previous step. Variables are set accordingly. The segment of code starts by looping upon the recordsets stored in the

opened query to perform a running total for a individual truck sub-classes (for multiple truck sub-classes, this loop is iterated with different variables).

Calculating the average empty operating weight for class *X* trucks.

```
Set dblowxx = CurrentDb()
Set rslowxx = dblowxx.OpenRecordset("Class X - xx LOW")

Dim AVEL As Currency
Dim AVELxx As Currency
Dim a As Integer

Let a = 0
Let AVELxx = 0
Do While Not rslowxx.EOF
a = a + 1
AVELxx = AVELxx + rslowxx![twt]
rslowxx.MoveNext
Loop
```

Perform weight average on low weight.

```
Dim AVELa As Currency
If a > 0 Then
AVEL = AVELxx / a
Else
AVEL = 0
End If
```

Perform weighted average on low weight.
AVEL*aa* is calculated the same way as AVEL*xx*, but new truck sub-class.
Count for new sub-class = *b*.

```
Dim AVELx, AVELy As Currency
If a > 0 Then
AVELa = AVELxx / a
Else
AVELa = 0
End If
If b > 0 Then
AVELb = AVELaa / b
Else
AVELb = 0
End If
If (a + b) > 0 Then
AVEL = ((AVELxx * a) + (AVELaa * b)) / (a + b)
```

Else
AVEL = 0
End If

6.) Next, the same procedure is followed as described in steps 3 and 4 to determine overweight truck information.

Set dbhigh*xx* = CurrentDb()
Set rshigh*xx* = dbhigh*xx*.OpenRecordset("Class *X* - *xx* HIGH")
Set dbclass*Xxx* = CurrentDb()
Set rsclass*Xxx* = dbclass*Xxx*.OpenRecordset("Class *X* - *xx*")

Dim *d* As Integer

Dim AVEOVE As Currency
Dim AVEOVE*xx* As Currency
Dim PEROVE As Currency
Dim PEROVE*xx* As Currency
Dim per As Currency
Dim per*xx* As Currency
Dim TOTAL As Currency
Dim TOTAL*xx* As Currency

Let *d* = 0
Let AVEOVE*xx* = 0
Do While Not rshigh*xx*.EOF
***d* = *d* + 1**
AVEOVE*xx* = AVEOVE*xx* + rshigh*xx*![twt]
rshigh*xx*.MoveNext
Loop

Perform weight average on over-weight.
If *d* > 0 Then
AVEOVE = AVEOVE*xx* / *d*
Else
AVEOVE = 0
End If

Perform weighted average on weight.
AVEL*aa* is calculated the same way as AVEL*xx*, but new truck sub-class.
Count for new sub-class = *y*.
Dim AVEOVE*d*, AVEOVE*e* As Currency
If *d* > 0 Then

```
AVEOVEd = AVEOVExx / d
Else
AVEOVEd = 0
End If
If e > 0 Then
AVEOVEe = AVEOVEaa / e
Else
AVEOVEe = 0
End If
If (d + e) > 0 Then
AVEOVE = ((AVEOVExx * d) + (AVEOVEaa * e)) / (d + e)
Else
AVEOVE = 0
End If
```

7.) The next code segment places information into variables for sending statistical weight information to "Class X Weight" table. Both single and multiple truck sub-class code procedures are shown.

```
If x > 0 Or a > 0 Or d > 0 Then
rsclassXxx.MoveLast
TOTALxx = rsclassXxx.RecordCount
Else
TOTALxx = 0
End If
```

With multiple truck sub-classes.

```
If y > 0 Or b > 0 Or e > 0 Then
rsclassAaa.MoveLast
TOTALaa = rsclassAaa.RecordCount
Else
TOTALaa = 0
End If
TOTAL = TOTALxx + TOTALaa

If d > 0 Then
perxx = rshighxx.RecordCount
Else
perxx = 0
End If
```

With multiple truck sub-classes.

```
If e > 0 Then
peraa = rshighaa.RecordCount
Else
```

```
perаа = 0
End If
per = perxx + (peraa)
```

8.) The following code segment sends information to the "Class *X* Weight" table in the 'WIM Data' database. The code also calls information from the "LEGAL", "LEGAL + HIGH", "HIGH" queries, and from the "wim data" table.

```
With rsstat
.AddNew
![AveOpWeight(kips)] = AVEOW
![AveLowWeight(kips)] = AVEL
If TOTAL > 0 Then
![%Overweight] = (per / TOTAL) * 100
Else
![%Overweight] = 0
End If
![AveOverweight(kips)] = AVEOVE
If d > 0 Then
![MaxOverweightxx(kips)] = DMax("[twt]", "Class X - xx LEGAL + HIGH")
Else
![MaxOverweightxx(kips)] = 0
End If
If x > 0 Then
![Aveweightxx] = DAvg("[twt]", "Class X - xx LEGAL")
Else
![Aveweightxx] = 0
End If
![Count] = DCount("[ca]", "Class X")
![Perc] = ((DCount("[ca]", "Class X")) / (DCount("[day]", "wim data")))
![CountHxx] = DCount("[twt]", "Class X - xx HIGH")
![CountLxx] = DCount("[twt]", "Class X - xx LEGAL")
![d1] = DFirst("[day]", "wim data")
![d2] = DLast("[day]", "wim data")
![m1] = DFirst("[month]", "wim data")
![m2] = DLast("[month]", "wim data")
![y1] = DFirst("[year]", "wim data")
![y2] = DLast("[year]", "wim data")
.UPDATE
End With
```

9.) The next segment of code sends information to the 'Annual Information' database. As initially done, a table needs to be opened for data storage, and variables need to be defined.

Dim dban*X* As DATABASE
Dim rsan*X* As Recordset

Set dban*X* = OpenDatabase("c:\my documents\Annual Info.mdb")
Set rsan*X* = dban*X*.OpenRecordset("Class *X* AW")

Dim AW*xx*J, AW*xx*F, AW*xx*MR, AW*xx*AP, AW*xx*MY, AW*xx*JN, AW*xx*JL, AW*xx*AU, AW*xx*S, AW*xx*O, AW*xx*N, AW*xx*D As Currency
Dim Count*xx*J, Count*xx*F, Count*xx*MR, Count*xx*AP, Count*xx*MY, Count*xx*JN, Count*xx*JL, Count*xx*AU, Count*xx*S, Count*xx*O, Count*xx*N, Count*xx*D As Currency
For multiple truck sub-classes.
Dim AW*aa*J ..... As Currency
Dim Count*aa*J ..... As Currency

10.) Next, values are calculated and sent to a table in the 'Annual Information' database for later usage in a function located in the 'Annual Information' database. The example below contains procedures for sending information of both a single truck sub-class, and multiple truck sub-classes using a weighted average to determine values.

With rsan*X*
.AddNew

Process copied for each month of the year using the variable dimensions above.
For single truck sub-class.
If *x*Jan > 0 Then
![Ca*X*AWJan] = TOTAL*xx*Jan / *z*Jan
![Count*X*Jan] = *z*Jan
Else
![Ca*X*AWJan] = 0
![Count*X*Jan] = 0
End If

For multiple truck sub-classes.
If *z*Jan > 0 Then
Aw*xx*J = TOTAL*xx*Jan / *z*Jan
Count*xx*J = *z*Jan
End If
If *y*Jan > 0 Then
Aw*aa*J = TOTAL*aa*Jan / *y*Jan
Count*aa*J = *y*Jan
End If
If (Count*xx*J + Count*aa*J) > 0 Then

![Ca*X*AWJan] = ((AW*xx*J * Count*xx*J) + (AW*aa*J * Count*aa*J)) / (Count*xx*J + Count*aa*J)
![Count*X*Jan] = Count*xx*J + Count*aa*J
End If
.UPDATE
End With

11.) The final step of the Weight code is to close all of the open tables and queries that have not been closed to this point. This is accomplished by the following command.

rsan*X*.Close