



**The I-69 Evansville-to-Indianapolis Study**  
**Tier 1 Environmental Impact Statement**

**Task 3.3.3.1 Technical Report**  
**Travel Model Users' Guide**

**March 7, 2001**



## 1.0 Introduction

This memorandum presents the Task 3.3.3 Travel Model Users' Guide for the I-69 Evansville to Indianapolis Tier 1 Environmental Impact Statement (I-69 Evansville to Indianapolis Study). This Users' Guide is intended to present detailed instructions for practitioners to use the updated Indiana Model for transportation planning applications for the I-69 Evansville to Indianapolis Study and other current and upcoming corridor projects for INDOT. This Users' Guide is also intended to be used in conjunction with the Task 3.3.3 Model Documentation Report that presents the full, detailed procedures of the updated Indiana Model.

The suite of analytical tools developed for the Major Indiana Corridor Investment Benefit Analysis System (MCIBAS), including the Indiana Statewide Travel Model (initial Indiana Model), formed the foundation for the updated Indiana Model developed specifically for the I-69 Evansville to Indianapolis Study. The initial Indiana Model, developed in April 1998, was updated to include the following revisions:

- **Increased Level of External-Indiana Transportation Network and Zone Detail.** The initial Indiana Model included limited transportation network and zone system detail in the states adjacent to Indiana. Typically, these out-of-state zones and network were configured with external stations. Detailed external-state (Illinois, Kentucky, Ohio, and Michigan) transportation network, zone system, and socioeconomic data were added to the modeling system to improve the internal-external travel movements within the updated Indiana Model. The four-step transportation models (trip generation, trip distribution, mode choice, and trip assignment), consistent with the internal elements of the Indiana Model, were used as the basis for estimating travel demand in each external state. This process ensures that the updated Indiana Model can be used to better identify the travel demand impacts associated with alternatives that have alignments adjacent to other states, and to better identify the internal-Indiana to external-Indiana (and vice-versa) movements. In addition, the updated Indiana Model will be able to identify the potential cumulative travel impacts associated with the multi-state National I-69 alternatives on the I-69 and other Indiana corridors.
- **Transportation Networks and Traffic Analysis Zones Refined and Updated.** The initial Indiana Model represented the major facilities of the statewide highway network and did not contain some important connectors and minor arterials significant to the connectivity of the southwestern Indiana transportation system. The zones represented in the initial Indiana Model also represented large geographic areas. Added transportation network detail was coded in the updated Indiana Model to better assess and identify the travel demand impacts of the preferred transportation alternatives. This process included converting the INDOT Road Inventory System to TransCAD as the basis for the updated 1998 transportation network. The 1998 Streets Dataset from Caliper Corporation containing the roadway attributes for functional classifications below State Routes was the primary source for increasing the level of network detail to the updated Indiana Model. In addition, a number of Traffic Analysis Zones were divided to cover smaller geographic areas. Note that these transportation network and zonal refinements were made with the Study Area only.



- **Other Transportation Network Detail.** Observed free-flow speed data collected specifically for the I-69 Evansville to Indianapolis Study were coded into the updated Indiana model. Traffic signalization information was coded into the network, and new peak period capacities were incorporated, based upon signalization and area type effects. This included adjusted travel times to reflect delays introduced by signalization. Note that the transportation network and zone system updates were made in the Study Area only.
- **Updated Socioeconomic Data to Reflect 1998 and 2025 Conditions.** Socioeconomic data representing 1998 conditions in Indiana and each adjacent state were coded into the updated Indiana Model zone system. These data were used as the basis for implementing and validating the four-step transportation models used to assess travel demand for the I-69 Evansville to Indianapolis Corridor Study. Future forecasts of socioeconomic data for 2025 were also input into the updated Indiana Model. Both 1998 and 2025 socioeconomic data were incorporated for the entire zone system in the updated Indiana Model. In addition, major post-1998 employment increases in the Study Area were identified, and these were added to 2025 employment projections, along with associated increases in population in the Study Area.
- **Models Based on Four-Step Transportation Model Principals.** The incremental modeling process that was implemented in the initial Indiana Model was refined to include the traditional four-step travel demand model validation procedures. Previous modeling elements such as the estimation and validation of base year origin and destination trip tables, the foundation for the incremental modeling process, were replaced with four-step model validation procedures including trip generation, trip distribution, mode choice, and trip assignment. Updating the Indiana Model using four-step modeling procedures simplifies the model application process and ensures that state-of-the-modeling practice is maintained.
- **Automation of the Updated Indiana Model.** The Geographic Information Systems Developers Kit (GISDK) is the programming language included with the TransCAD software package. TransCAD provides all the tools and commands necessary to run the four-step transportation modeling process through its menus and graphical user interfaces (GUIs). GISDK allows the user to create programs to automate these processes and provide enhanced GUIs for simplified operations within the travel modeling environment. It is also possible to automate multiple procedures for a single step in the travel modeling chain. For example, when trip distribution is executed, a number of steps are conducted such as balancing productions and attractions, creating transportation networks, running network travel time skims, and manipulating matrices. By implemented specific GISDK within the updated Indiana Model, multiple processes can be automated to execute with a single mouse click, whereas executing through the standard GUI is much more cumbersome. Using GISDK provides INDOT with a very powerful tool that allows the user to run many alternatives with the same settings every time, and reduces the amount of set-up time that would typically be required through the implementation of the standard TransCAD GUI. Appendices B & C contain the code for these GISDK programs; these appendices are NOT included with most copies of this report. These programs previously have been provided to INDOT's technical staff.
- **Emphasis of Calibration and Validation.** The calibration and validation efforts were undertaken primarily to ensure adequate model performance within the Study Area. Model performance statistics from outside of the study area also were analyzed, but primary emphasis was given to model performance within the Study Area.

Detailed descriptions of each modeling element summarized above will be presented in the Task 3.3.3 Model Documentation Report.



The Users' Guide is structured to identify the application methods of the core modeling components of the updated Indiana Model including very detailed instructions on installing and running the modeling system presented in Section 2.0. Section 3.0 presents the application procedures to run trip generation and Section 4.0 presents the detailed descriptions for running trip distribution. Sections 5.0 and 6.0 present the mode choice and time of day modeling procedures while Section 7.0 presents the procedures to run the trip assignment process.



## **2.0 Model Installation and Set-Up**

The following procedures should be used to install and set-up the updated Indiana Model system. The software and hardware system requirements, directory structure, and procedures required to start the TransCAD and GISDK programs are presented in this section.

### **■ 2.1 System Requirements**

The minimum software and hardware system requirements recommended for the proper execution of the updated Indiana Model include:

#### *Software*

Microsoft Windows NT  
TransCAD Version 3.61

#### *Hardware*

PentiumII 500  
128MB RAM – Minimum  
1GB – minimum of free hard disk space, each alternative is 750MB complete

### **2.2 System Directory Structure**

The directory structure provided in this section is required for the updated Indiana Model system to run properly. This structure was established for purposes of evaluating multiple scenarios and should be stored based on the root directory established by the analyst. Table 2.1 illustrates the directory structure that should be used to install and run the updated Indiana Model. The subdirectories below the root directory (in this case '98base') are mandatory. Analysts can establish additional directories using the same structure with a different model root directory, e.g., 2020\_no\_build and 2020\_build\_alternative1.



**Table 2.1 Updated Indiana Model Directory Structure**

C:\your_directory\98base\	---	Model root directory
hwy\	---	Line geography file
mc\	---	Mode choice files
taz\	---	Transportation analysis zone boundary layer
td\	---	Trip distribution files
tg\	---	Trip generation files
tod\	---	Time of day files
Routes\	---	TransCAD route system files (for use in mode choice analysis and transit building)

Source: Cambridge Systematics, Inc.

The subdirectories consider the separate data and modeling elements of the updated Indiana Model. For example, the hwy and taz subdirectories contain the transportation network and zone system layers and other TransCAD files associated with each scenario. The mc, td, tg, and tod subdirectories represent the modeling elements of the system including mode choice, trip distribution, trip generation, and time of day. The routes subdirectory contains a file to generate the route system, the transit network, and the transit skim used in mode choice modeling.

## ■ 2.3 Starting the TransCAD and GISDK Program

To run the updated Indiana Model, analysts need to start TransCAD v 3.61.

### Open Line Layer

Analysts should first load the geographic line layer onto a map within the TransCAD environment. To do this, follow the steps identified below:

1. File → Open;
2. Specify geographic file as the file type;
3. Navigate to the \your\_directory\98base\hwy subdirectory;
4. Select the appropriate geographic line layer for the alternative and click OK; and
5. A map of the geographic line layer will open.



### Open Zone Layer

Next the analyst should load the TAZ layer onto the map previously opened (see above). The steps identified below should be followed to complete this process:

1. Map → Layers; this will open the add layers dialog for the current map;
2. Choose add layer from the dialog;
3. Navigate to the \your\_directory\98base\taz subdirectory;
4. Select the TAZ layer;
5. Close the Layers dialog; and
6. The selected TAZs will now be opened and loaded on the map.

### Install GISDK Program



When the geographic files have been successfully loaded, install the GISDK add-in. To do this, follow the steps below:

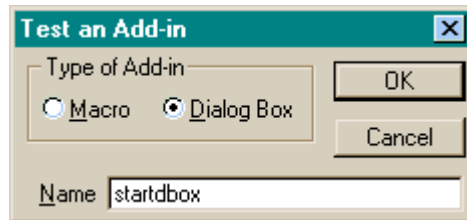
1. Tools → Add-ins;
2. Highlight GIS developers kit and click OK;



3. The dialog box below will open;



4. Using the compile button from the GISDK toolbox,  compile the resource file either “dkmodel4\_98.rsc” or “dkmodel4\_25.rsc” by selecting it and click open;
5. Run the program using the Test Add-in button ;
6. The Test Add-in button will open a dialog box below; and



7. Click on the ‘Dialog Box’ radio button and enter “startdbox” in the ‘Name’ textline as shown above.

### The Modeling Dialog Box

At the end of the above process, the modeling dialog box shown in Figure 2.1 will open. This dialog box is the primary control of the updated Indiana Model system. On the left side of the dialog box in the ‘General Settings’ frame, the line layer, node layer, and TAZ layer must be specified to run correctly. The highest zone numbers for Indiana, External areas, and total zones must be specified and must be reflected in the TAZ layer id field (see Appendix A for a summary of the TAZ layer database definitions). In the current model the highest zone numbers are, internal 742, external 1113, and total zones 2126. In addition, the modeling path or root directory needs to be specified as presented in Section 2.2 of this memorandum. The user should use the browse button to identify the appropriate modeling root directory.

The buttons on the right side of the dialog box shown in Figure 2.1 allow the analyst to execute each modeling step separately. The ‘Run All Steps’ button can be used to run the complete modeling system in its entirety. This means the process of Trip Generation through Trip Assignment can be fully automated and run without stopping.

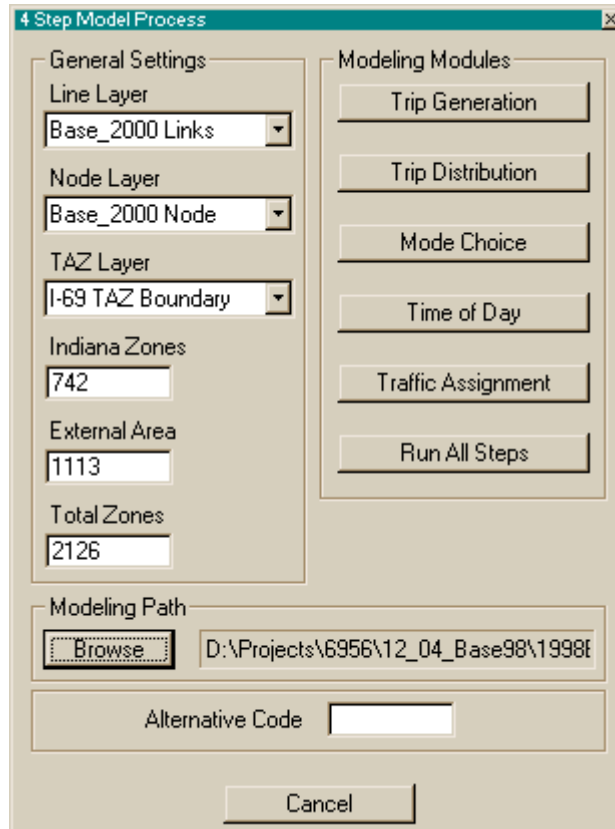
The alternative code box shown at the bottom of the dialog box in Figure 2.1 has not been implemented and does not need to be filled in. All other settings in the Modeling dialog box must be specified for the updated Indiana Model to run correctly. It is important to note that users must adhere to these guidelines. For example, several input and output parameters are passed between each modeling step not only from the dialog box, but also from particular modules that depend on running prior modeling steps (e.g., trip distribution cannot be run until trip generation is completed). The



subsequent sections documented in this memorandum refer to input files needed for the proper execution of the updated Indiana Model. It is important to note that only the geographic files should be loaded in TransCAD. All other input files need to exist at the specified location, but should not be open in TransCAD at the time of execution. If they are open in TransCAD the model will not run correctly.



Figure 2.1 Modeling Dialog Box



Source: Cambridge Systematics, Inc.



## **3.0 Trip Generation**

The procedures used to run the trip generation element of the updated Indiana Model are presented in this section. Internal-Indiana and external-to-Indiana (Illinois, Kentucky, Michigan, and Ohio) trip productions and attractions are generated in this process. The required input files and programs, TransCAD program macros, and outputs generated in this process are presented below.

### **■ 3.1 Required Input Files and Programs**

Running the trip generation program macros for the updated Indiana Model requires the following data inputs:

- Geographic line layer loaded in the current TransCAD Session;
- Node layer of the Geographic Line Layer;
- TAZs loaded in the current TransCAD Session;
- Household size and auto ownership percentages of each internal and external TAZ;
- Statewide control percentages for household size and auto ownership;
- Home-Based Work production rates by super zone for internal and external TAZs;
- Home-Based Other production rated by super zone for internal and external TAZs;
- Non-Home Based production rates by super zone for internal and external TAZs;
- Long Trip production rates by super zone for internal and external TAZs; and
- Percentage of trip making inside Indiana by trip purpose and super zone.

The file names for these required input files are presented for each macro described in Section 3.2. Refer to Appendix A: that shows the detailed descriptions and field definitions for each input variable in the TransCAD database files of the updated Indiana Model.



## ■ 3.2 Trip Generation TransCAD Macros

When the user selects the trip generation button on the modeling dialog box, three TransCAD macros are used for trip generation in the updated Indiana Model. First, “create\_hh\_dist” is run to create the household and auto ownership distributions for each zone for direct input into the trip generation process. Second, “create\_prods” is run to generate the productions for each zone, and third, the “attraction\_model” is executed to generate attractions for each zone. All input and output files are stored in the \your\_directory\98base\tg. Each macro is described in detail in the following section.

### ***Trip Generation Macro “create\_hh\_dist”***

This macro has a number of required inputs necessary for proper execution. Table 3.1 shows the required input files and Table 3.2 presents the results and household distribution outputs after this program is executed.

**Table 3.1 Required Input Files to Run Macro "create\_hh\_dist"**

Input Files	Description	File Location
Pcts_exp.dbf	Household size and auto ownership percentages of each internal TAZ. See Appendix A for field definitions. This file is created externally.	\your_directory\98base\tg
Swtpct.dbf	Statewide control percentages for household size and auto ownership. This file is created externally.	\your_directory\98base\tg

Source: Cambridge Systematics, Inc.

**Table 3.2 Household Distribution Output File Generated by "create\_hh\_dist"**

Output Files	Description	File Location
“final_hh.dbf”	Final household distribution file and is passed to Create Prods Macro	\your_directory\98base\tg

Source: Cambridge Systematics, Inc.



**Trip Generation Macro “create\_prods”**

This macro calculates the trip productions for each TAZ. The macro is called from “create\_hhdist” macro. Table 3.3 shows the required input files needed for proper execution. All files are stored in the \your\_directory\98base\tg subdirectory of the model root directory. Table 3.4 shows the output file name for the trip productions and attractions.

**Table 3.3 Required Input Files to Run Macro "create\_prods"**

Input Files	Description	File Location
Hbwprate_30.dbf	Home-Based Work production rates by super zone for internal zones. Created externally.	\your_directory\98base\tg
Hboprte_30.dbf	Home-Based Other production rates by super zone for internal zones. Created externally.	\your_directory\98base\tg
Nhbprate_30.dbf	Non-Home Based production rates by super zone for internal zones. Created externally.	\your_directory\98base\tg
Lngprate_30.dbf	Long Trip production rates by super zone for internal zones. Created externally.	\your_directory\98base\tg
Intpcts_exp.dbf	Percentage of trip making inside Indiana by purpose and superzone. Created externally.	\your_directory\98base\tg

Source: Cambridge Systematics, Inc.

**Trip Generation Macro “attraction\_model”**

This macro creates the trip attractions for each TAZ in the updated Indiana Model. The macro is initiated from the “create\_hhdist” macro. This Macro uses files generated in the above procedure for trip productions. The input parameters are integrated and passed to this macro internally to TransCAD; therefore, no input file needs to be called for by the user. As with the productions, Table 3.4 shows the attractions generated through this modeling process. The pre-balanced productions and attractions computed as part of the trip generation process are automatically attached to the TAZ layer for further use in the Indiana Model stream.



**Table 3.4 Results and Output File Generated From Macro "attraction\_model"**

Output Files	Description	File Location
TAZ Layer	Pre-balanced Productions and Attractions are stored on this layer and used throughout subsequent macros.	\your_directory\98base \taz

Source: Cambridge Systematics, Inc.



## 4.0 Trip Distribution

The procedures used to run the trip distribution element of the updated Indiana Model are presented in this section. As with trip generation, internal-Indiana and external-to-Indiana (Illinois, Kentucky, Michigan, and Ohio) trip distribution are generated in this process. The required input files and programs, TransCAD program macros, and outputs generated in this process are presented below. Users should make sure that the trip generation procedures identified in Section 3.0 should be run before starting trip distribution.

### 4.1 Required Input Files and Programs

Running the trip distribution procedures of the updated Indiana Model requires the following data inputs:

- Geographic line layer loaded in the current TransCAD Session;
- Node layer of the Geographic Line Layer;
- TAZs loaded in the current TransCAD Session;
- Highest TAZ-ID of internal and external TAZs entered in the Dialog Box;
- Externally created KFactor Matrix; and
- Externally created Friction Factor database file.

### 4.2 Trip Distribution TransCAD Macros

When a user selects the trip distribution button on the modeling dialog box, five macros are used to run trip distribution of the updated Indiana Model. First, "balance\_ps\_to\_as" is used to balance the productions and attractions created in the previous trip generation modeling steps. Second, "create\_hnet" is run to create a highway network required for travel time skimming. Third, "Multi\_Path" macro is run to create transportation network travel time skims for internal and external zones. This macro is a batch script generated by TransCAD. Fourth, "friction\_factors" macro is run to generate friction factor matrices to be used to run the fifth trip distribution macro, "TripDist". These macros, with the exception of production and attraction balancing macro ("balance\_ps\_to\_as"), are executed twice, first for internal zones and second for external zones. All input files are retrieved from the \your\_directory\98base\td, except highway networks generated in the "create\_hnet" macro (found in \your\_directory\98base\hwy).

#### ***Trip Distribution Macro "balance\_ps\_to\_as"***

This macro balances the productions and attractions generated in the trip generation process presented in Section 3.0. Only input parameters from the program and modeling dialog box are needed for execution. The balanced productions and attractions replace the pre-balanced values in the TAZ layer.



### ***Trip Distribution Macro “create\_hnet”***

This macro creates multiple highway networks used to generate highway skims in the gravity model, trip distribution process developed for the updated Indiana Model. First the macro is executed for the internal-Indiana TAZ’s only, and subsequently run again for the external-to-Indiana areas in Illinois, Kentucky, Michigan, and Ohio. Table 4.1 shows the output networks created through this process. The networks are stored in your\_directory\98base\hwy.

**Table 4.1 Output Transportation Networks Generated by "create\_hnet"**

Output Files	Description	File Location
Highway_int.net	Internal highway network used for internal zone to zone skims for long trip purpose.	your_directory\98base\hwy
Highway_intext.net	Internal/External highway network used for internal – external zone to zone skims for all other purposes.	your_directory\98base\hwy

Source: Cambridge Systematics, Inc.

### ***Trip Distribution Macro “Multi\_Path”***

This macro creates an impedance matrix of free flow travel time (FFTIME), length, and congested travel time (CTIME) using the transportation networks generated using the previous macro. The process is executed twice, once for internal-Indiana zones and again for external-to-Indiana zones. There are a number of internal parameters passed to this macro from the dialog box and previous macros in trip distribution, therefore no input files are needed. Table 4.2 shows the output files that are generated. The macro is generated using TransCAD batch macro process and modified specifically for the updated Indiana Model.



**Table 4.2 Output Impedance Matrices Generated by "multi\_path"**

Output Files	Description	File Location
Imp01.mtx	Internal zone to zone impedance matrix used for long trip purpose.	your_directory\98base\ td
Imp02.mtx	Internal/External zone to zone impedance matrix used for all other purposes.	your_directory\98base\ td

Source: Cambridge Systematics, Inc.

***Trip Distribution Macro "friction\_factors"***

This macro creates two friction factor matrices. The first matrix is created for distributing the long trip purpose, and the second is used for distributing trips for all other purposes, home-based work, home based other, and non-home based. Table 4.3 presents the required input files necessary for execution and Table 4.4 shows the resulting output files. All files are stored in your\_directory\98base\td subdirectory.

**Table 4.3 Required Input Files to Run "friction\_factors"**

Input Files	Description	File your_directory\98base \td Location
Ffactors.bin	Binary data file of friction factors lookup table. See appendix A for definition. Created externally.	Your_directory\98base \td
Ffactors.dcb	Dictionary file for above file. Created externally.	Your_directory\98base \td

Source: Cambridge Systematics, Inc.



**Table 4.4 Results and Output File Generated by "friction\_factors"**

Output Files	Description	File Location
ffact01.mtx	Final friction factor matrix created for long trip purpose from Tempimp01.bin	Your_directory\98base \td
ffact02.mtx	Final friction factor matrix created for Home-Based Work, Home-Based Other, and Non-Home Based purposes from Tempimp02.bin	Your_directory\98base \td

Source: Cambridge Systematics, Inc.

***Trip Distribution Macro "TripDist"***

This macro runs the gravity model and distributes trips for both internal-Indiana and external-to-Indiana area. This macro was created using TransCAD batch processing routines. The gravity model is run twice, once for the internal-Indiana areas, and again for the external-to-Indiana areas. Table 4.5 shows the required files need for proper execution and Table 4.6 shows the outputs after execution.

**Table 4.5 Required Input Files to Run "TripDist"**

Input Files	Description	File Location
Ffact01.mtx	Friction Factor Matrix for Long Trip Purpose (Created from friction_factors macro)	Your_directory\98base \td
Ffact02.mtx	Friction factor matrix for Home-Based Work, Home-Based Other, and Non-Home Based. (Created from friction_factors macro)	Your_directory\98base \td
Kfactor.mtx	Kfactor matrix created externally.	Your_directory\98base \td

Source: Cambridge Systematics, Inc.



**Table 4.6 Results and Output File generated from "TripDist"**

Output Files	Description	File Location
Ptripdist01.mtx	Person trip tables for long trip purpose.	Your_directory\98base \td
Ptripdist02.mtx	Person trip tables for home-based work, home based other and non-home based purposes.	Your_directory\98base \td

Source: Cambridge Systematics, Inc.



## 5.0 Mode Choice

The procedures used to run the mode choice elements of the updated Indiana Model are presented in this section. As with the previous modeling steps, internal-Indiana and external-to-Indiana (Illinois, Kentucky, Michigan, and Ohio)

The required input files and programs, TransCAD program macros, and outputs generated in this process are presented below. Users should make sure that the trip generation and trip distribution procedures identified in Sections 3.0 and 4.0 are run before starting mode choice.

### 5.1 Required Input Files and Programs

Running the updated Indiana Model mode choice program macros requires the following data inputs:

- Impedance matrix created from trip distribution;
- Transit skim created in TransCAD externally of the modeling programs;
- Binary modeling file created externally of the modeling macros;
- Person trip tables created from trip distribution; and
- Corridor 18 external trip matrix created externally of the program macros.

### 5.2 Mode Choice TransCAD Macros

When a user selects the mode choice button on the modeling dialog box, three macros make up the mode choice element of the updated Indiana Model. In addition to the files specified in the previous sections of this memorandum, there are a number of files created from each macro that are used as inputs into mode choice. First, "thematrix" performs various matrix manipulations to create time and cost matrices. Second, "mnl-eval-Ing" executes the multinomial logit model for long trips internally in Indiana. Third, "test\_matrix" creates purpose based vehicle trips tables to be used in the time of day macros.



**Mode Choice Macro “thematrix”**

This macro creates time, cost, and transit cost matrices used for mode choice modeling input. Table 5.1 shows the required input matrices, the required matrix core names, and the subdirectory where the files are located. Table 5.2 shows the outputs and results of this macro.

**Table 5.1 Required Input Matrices to Run "thematrix"**

Input Matrix	Required Matrix Cores	File Location
Imp01.mtx	FFTIME (Create from Trip Distribution)	Your_directory\98base\td
Tr_Skim.mtx	FFTIME (non-transit), FFTIME(in-vehicle), CTIME (non-transit), CTIME(in-vehicle) Created Externally	Your_directory\98base\routes

Source: Cambridge Systematics, Inc.

**Table 5.2 Output Matrices Created by "thematrix"**

Input Matrix	Matrix Cores Created	File Location
Time.mtx	FFLOW*, CTIME, FFTRAN_IVTT, FFTRAN_OVTT, CTTRAN_IVTT, CTTRAN_OVTT	Your_directory\98base\mc
Cost.mtx	Length (Transit), Auto, Transit	Your_directory\98base\mc

Source: Cambridge Systematics, Inc.

**Mode Choice Macro “mnl\_eval\_lng”**

This is a customized TransCAD batch macro specifically developed for the updated Indiana Model. The macro runs the multinomial logit model for long trips. Table 5.3 shows the required input files and matrices. Table 5.4 shows the outputs generated from this macro.



**Table 5.3 Required Input Files and Matrices to Run "mnl\_eval\_lng"**

Input Matrix	Required Matrix Cores	File Location
Lngmodel.bin	Not a Matrix (Created externally see TransCAD manual for details)	Your_directory\98base \mc
Lngmodel.dcb	Dictionary for above file.	Your_directory\98base \mc
Time.mtx	FFTAN_IVTT (Created by thematrix macro)	Your_directory\98base \mc
Cost.mtx	Auto IVTT (Created by thematrix macro)	Your_directory\98base \mc

Source: Cambridge Systematics, Inc.

**Table 5.4 Output Matrix Created by "mnl\_eval\_lng"**

Input Matrix	Matrix Cores Created	File Location
Lngshares.mtx	Auto, LNG	Your_directory\98base \mc
Cost.mtx	Length (Transit) ), Auto, Transit	Your_directory\98base \mc

Source: Cambridge Systematics, Inc.

**Mode Choice Macro "test\_matrix"**

This macro performs a variety of TransCAD matrix manipulations to create purpose based vehicle trip tables for input into the time of day element of the updated Indiana Model. Table 5.5 shows the required input matrices required for proper execution. It should be noted that the matrix core required for the corr18lng.mtx varies by analysis year. Table 5.6 shows the final vehicle trip matrix that is created through this process.



**Table 5.5 Required Input Matrices to Run "test\_matrix"**

Input Matrix	Required Matrix Cores	File Location
Prtripdist02.mtx	HBW, HBO, NHB (Created in Trip Generation)	Your_directory\98base \td
Prtripdist01.mtx	LNG (Created in Trip Generation)	Your_directory\98base \td
Lngshares.mtx	Auto (Created in mnl_eval_lng macro)	Your_directory\98base \mc
Corr18_lng.mtx	CAR98_LNG – For 1998 Model CAR25_LNG – For 2025 Model (Created externally)	Your_directory\98base \mc

Source: Cambridge Systematics, Inc.

**Table 5.6 Output Matrices Created by "test\_matrix"**

Output Matrix	Matrix Cores Created	File Location
Hbwbase.mtx	Auto, Transit, HBW_VEH_TRP	Your_directory\98base \mc
Hbobase.mtx	Auto, Transit, HBO_VEH_TRP	Your_directory\98base \mc
Nhbbase.mtx	Auto, Transit, NHB_VEH_TRP	Your_directory\98base \mc
Lngbase.mtx	Auto, LNG_VEH_TRP	Your_directory\98base \mc

Source: Cambridge Systematics, Inc.

## 6.0 Time of Day

The procedures used to run the time of day element of the updated Indiana Model are presented in this section. As with the previous modeling steps, separate internal-Indiana and external-to-Indiana (Illinois, Kentucky, Michigan, and



Ohio) time of day estimates are generated in this process. The required input files and programs, TransCAD program macros, and outputs generated in this process are presented below. Users should make sure that the previous modeling procedures identified in Sections 3.0 through 5.0 are run before starting time of day.

■ **6.1 Required Input Files and Programs**

Running the updated Indiana Model time of day program macros requires the following data inputs:

- Base vehicle trip matrix for each purpose (HBW, HBO, NHB and LNG) created from the mode choice macros;
- Corridor 18 external trip matrix created externally of the program macros; and
- Truck matrix created externally using TransCAD ODME process.

■ **6.2 Time of Day TransCAD Macro**

When a user selects the time of day button on the modeling dialog box, one macro is initiated to conduct the time of day element of the updated Indiana Model. In addition to the files specified in the previous sections of this memorandum, there are a number of files created from each macro that are used as inputs into mode choice. The "matrix\_manip" macro performs various matrix manipulations to apply the time of day factors generated from the 1995 Indiana Household Travel Survey.

***Time of Day Macro "matrix\_manip"***

This macro creates the final time of day matrices, representing the morning (a.m.) and afternoon (p.m.) peak periods and the off peak period (off), used in the trip assignment element of the Indiana Model for both trucks and autos. The time of day, production and attraction factors generated from the 1995 Indiana Household Travel Survey, are hard coded into the macro. The only passed parameter input called for in this macro is the modeling path specified in the Dialog Box. The corr18\_lng.mtx requires different matrix core names for the analysis year, while the truck trip tables also vary by analysis year. These and all other required inputs are shown in Table 6.1. Table 6.2 shows the output matrices that are created and used in the assignment process.

**Table 6.1 Required Input Matrices to Run "matrix\_manip"**

Input Matrix	Required Matrix Cores	File Location
Hbwbases.mtx	HBW_VEH_TRP (Created in Mode Choice)	Your_directory\98base \mc
Hbobases.mtx	HBO_VEH_TRP (Created in Mode Choice)	Your_directory\98base \mc



	(Created in Mode Choice)	\mc
Nhbbase.mtx	NHB_VEH_TRP (Created in Mode Choice)	Your_directory\98base \mc
Lngbase.mtx	LNG_VEH_TRP (Created in Mode Choice)	Your_directory\98base \mc
Corr18_lng.mtx	CAR98_LNG – <b>For 1998 Model</b> CAR25_LNG – <b>For 2025 Model</b> (Created externally)	Your_directory\98base \mc
Odme_od13.mtx	ODME – <b>For the 1998 Model</b> (Created externally using TransCAD ODME process.)	Your_directory\98base \hwy
Corr18_25_Trk.mtx	Truck2025 – <b>For the 2025 Model</b> (Created externally using TransCAD ODME process.)	Your_directory\98base \hwy

Source: Cambridge Systematics, Inc.



**Table 6.2 Output Matrices Created by "matrix\_manip"**

Output Matrix	Matrix Cores Created	File Location
Todcorr18.mtx	Final AM Peak, Final PM Peak, Final OFF Peak (Used in assignment process)	Your_directory\98base\tod
todtruck.mtx	AM Peak, PM Peak, OFF Peak	Your_directory\98base\tod

Source: Cambridge Systematics, Inc.



## 7.0 Trip Assignment

The procedures used to run the trip assignment element of the updated Indiana Model are presented in this section. The trip assignment process covers the entire modeling area. The required input files and programs, TransCAD program macros, and outputs generated in this process are presented below. Users should make sure that the previous modeling procedures identified in Sections 3.0 through 6.0 are run before starting the trip assignment process.

### 7.1 Required Input Files and Programs

Running the trip assignment for the updated Indiana Model requires the following data inputs:

- Geographic line layer loaded in the current TransCAD Session
- Node layer of the Geographic Line Layer
- Transportation Analysis Zones loaded in the current TransCAD Session
- Base vehicle trip matrix for each purpose (HBW, HBO, NHB and LNG) created from the mode choice macros;
- Time of Day vehicle trip table matrix created from the mode choice macro; and
- Time of day truck matrix created from mode choice section from choice macro.

### 7.2 Trip Assignment TransCAD Macros

The assignment macros are run for each time period - a.m. peak, p.m. peak, and off peak. When a user selects the trip assignment button on the modeling dialog box, five macros are run during the process. First, "create\_hnet" is run to generate the transportation network for travel demand modeling. Second, "truckassign" is run to assign the truck trips by time of day to the network. (The truck trip tables were estimated using separate procedures from the four-step transportation models estimated for passenger vehicles. See the Task 3.3 Model Documentation Report for detailed information about truck modeling procedures used to support the Indiana Model.) Third, "create\_hnet" is run again to create the highway network with pre-loaded truck trips. Fourth, "assign" is a modified TransCAD batch macro and is used to assign all vehicle trips, except trucks, to the highway network for all time periods. Fifth, "finalcalcs" is a post processor and is run to add vehicle volumes and truck volumes by time period after the trip assignment process.

#### *Trip Assignment Macro "create\_hnet"*

This macro creates the highway network for the truck assignment process. Table 7.1 shows the output networks created. All transportation networks are stored in the Your\_directory\98base\hwy directory.

**Table 7.1 Output Networks Generated from "create\_hnet"**



Output Files	Description	File Location
Highway_intext.net	Highway network used for truck assignment process. Uses all zones in the model system.	Your_directory\98base\hwy

Source: Cambridge Systematics, Inc.

***Trip Assignment Macro "truckassign"***

This macro assigns the truck trips by time of day to the transportation network. It uses the time of day truck trip matrix created from the mode choice element and the highway network created in the previous macro. It is a TransCAD batch macro modified for use in the updated Indian Model. The macro is executed for each time period. Table 7.2 shows the required files required to run this macro, and Table 7.3 shows the output generated after execution. All assignment output files are stored in the \Your\_directory\98base\hwy.

**Table 7.2 Required Input Matrices to Run "truckassign"**

Input Matrix	Required Matrix Cores	File Location
Todtruck.mtx	AM Peak, PM Peak, OFF Peak (Created from mode choice)	Your_directory\98base\hwy

Source: Cambridge Systematics, Inc.



**Table 7.3 Output Trip Assignment Results Generated from "truckassign"**

Output Files	Description	File Location
Truckampeak_asn_link. bin	Truck assignment results for the AM Peak period.	Your_directory\98base \hwy
Truckpmpeak_asn_link .bin	Truck assignment results for the PM Peak period.	Your_directory\98base \hwy
Truckoffpeak_asn_link. bin	Truck assignment results for the OFF Peak period.	Your_directory\98base \hwy

Source: Cambridge Systematics, Inc.

***Trip Assignment Macro "create\_hnet" (second use of this Macro in this process)***

This macro is the same as the other "create\_hnet" described previously in this user guide with the exception that the truck volumes generated from the above macro are passed and used as pre-loads in the highway network generation process. This network with truck pre-loads is then passed to the next macro, "assign".

***Trip Assignment Macro "assign"***

This is a modified TransCAD batch macro and is used to assign all vehicle trips, except trucks, to the highway network for all time periods. Table 7.4 shows the required input files and the generated output files are shown in Table 7.5. All output files from this macro are stored in \Your\_directory\98base\hwy.

**Table 7.4 Required Input Matrices to Run "assign"**

Input Matrix	Required Matrix Cores	File Location
Todcorr18.mtx	AM Peak, PM Peak, OFF Peak (Created from mode choice)	Your_directory\98base \hwy

Source: Cambridge Systematics, Inc.



**Table 7.5 Output Trip Assignment Results Generated from "assign"**

Output Files	Description	File Location
Finalampeak_asn_link. bin	Vehicle assignment results for the AM Peak period.	Your_directory\98base \hwy
Finalpmpeak_asn_link. bin	Vehicle assignment results for the PM Peak period.	Your_directory\98base \hwy
Finaloffpeak_asn_link. bin	Vehicle assignment results for the OFF Peak period.	Your_directory\98base \hwy

Source: Cambridge Systematics, Inc.

### ***Trip Assignment Macro "finalcalc"***

This macro is a post processor with the only parameter required as input being the line view. This macro adds vehicle volumes and truck volumes by time period and truck. The results are daily auto volumes, daily truck volumes, and daily total volumes. The outputs are expressed as fields on the geographic line layer.



## **8.0 Run All Modeling Steps**

The run all steps button shown in the dialog box (see Figure 2.1) executes all steps of the updated Indiana Model from trip generation through the assignment process. Before the "Run All Steps" button can be executed, all information must be specified in the dialog box, and all input files for macro execution must be specified in appropriate subdirectories.

### **Appendix A - Detailed Indiana Model Field Names and Definitions**

The attached tables identify the field names and detailed descriptions for a variety of TransCAD files contained in the updated Indiana Model including:

- Tables A.1 and A.2 - Geographic File Definitions;
- Table A.3 through Table A.8 - Trip Generation Data File Definitions;
- Tables A.9 and A.10 - Trip Distribution Data File Definitions and Data Dictionaries; and
- Table A.11 - Mode Choice Data File Definitions and Data Dictionaries.



**Table A.1 Field Descriptions in the Southwest Indiana Model Base Year Roadway Database File:  
*i69\_base\_2000.dbd***

<b>Field Name</b>	<b>Field Description</b>
ID	Link ID
Length	Link Distance
Dir	Link Directionality
RTE	Route Name
AADT	Average Annual Daily Traffic
Adj_AADT	Adjusted Average Annual Daily Traffic
Truck_AADT_98	1998 Daily Truck Traffic
Posted_Speed	Posted Speed
FFSpeed	Free Flow Speed
Lanes	Number of Lanes - 2 Way
LN1DIR	Number of Lanes - 1 Way
LN_Width	Lane Width
PKhrCAP	Peak Hour Capacity
[Study Area]	Study Area Flag
FFTime	Free Flow Link Travel Time
Ctime	Congested Link Travel Time
FHWA_FC	FHWA Functional Class
AB_CAPPEAK	A to B Peak Period Capacity
BA_CAPPEAK	B to A Peak Period Capacity
AB_CAP	A to B Daily Capacity
BA_CAP	B to A Daily Capacity
[24hrCAP]	Daily 2-Way Capacity
SZ	Superzone
Finalcnt	Smoothed Average Daily Traffic
AM_TRK_VOL	AM Peak Period Truck Volume
AM_AUTO_VOL	AM Peak Period Auto Volume
PM_TRK_VOL	PM Peak Period Truck Volume
PM_AUTO_VOL	PM Peak Period Auto Volume
OP_TRK_VOL	OP Peak Period Truck Volume
OP_AUTO_VOL	OP Peak Period Auto Volume
Dly_TRK_VOL	Daily Truck Volume
Dly_AUTO_VOL	Daily Auto Volume
Dly_Tot_VOL	Daily Total Volume

**Source:** Cambridge Systematics, Inc.



**Table A.2 Field Descriptions in the Southwest Indiana Model Base Year Traffic Analysis Zone (TAZ)**  
**Database File: i69\_taz\_2000.bin**

<b>Field Name</b>	<b>Field Description</b>
ID	Node ID
Area	Land Area of TAZ
Taz	Traffic Analysis Zone Number ID used in GISDKprogram
SZ	Superzone Number
Households	Number of Households
ZFARMIND	Number of Farming and Industrial Employees
ZNONINDG	Number of Government and Non-Industrial Employees
ZIND	Number of Industrial Employees
ZNONIND	Number of Non-Industrial Employees
ZNUMEMP	Number of Emploeyss - All Types
ZGOVT	Number of Government Employees
HBW_PROD	Home-Based Work Productions*
HBO_PROD	Home-Based Other Productions*
NHB_PROD	Non-Home Based Productions*
LNG_PROD	Long Trip Productions*
HBW_ATTR	Home-Based Work Attractions*
HBO_ATTR	Home-Based Other Attractions*
NHB_ATTR	Non-Home Based Attractions*
LNG_ATTR	Long Trip Attractions*
ApproxRadius	Approximate Radius of TAZ
Pop	Population
IntraTT	Intrazonal Travel Time
Ext_SZ	External Area Superzone Number

**Source:** Cambridge Systematics, Inc.



**Table A.3 Field Descriptions – Trip Generation Household Size Distribution Database File:  
*pcts\_exp.dbf***

Field Name	Field Description
TAZ_ID1	Traffic Analysis Zone Number
PCTHHS1	Percentage of 1 Person Households
PCTHHS2	Percentage of 2 Person Households
PCTHHS3	Percentage of 3 Person Households
PCTHHS4	Percentage of 4+ Person Households
PCTAO0	Percentage of 0 Auto Households
PCTAO1	Percentage of 1 Auto Households
PCTAO2	Percentage of 2 Auto Households
PCTAO3	Percentage of 3+ Auto Households

Source: Cambridge Systematics, Inc.

**Table A.4 Field Descriptions – Statewide Household Demographic Percentages Database File:  
*swtpct.dbf***

Field Name	Field Description
HHS1A00	Statewide Percentage of 1 Person Households with 0 Autos
HHS1A01	Statewide Percentage of 1 Person Households with 1 Autos
HHS1A02	Statewide Percentage of 1 Person Households with 2 Autos
HHS1A03	Statewide Percentage of 1 Person Households with 3+ Autos
HHS2A00	Statewide Percentage of 2 Person Households with 0 Autos
HHS2A01	Statewide Percentage of 2 Person Households with 1 Autos
HHS2A02	Statewide Percentage of 2 Person Households with 2 Autos
HHS2A03	Statewide Percentage of 2 Person Households with 3+ Autos
HHS3A00	Statewide Percentage of 3 Person Households with 0 Autos
HHS3A01	Statewide Percentage of 3 Person Households with 1 Autos
HHS3A02	Statewide Percentage of 3 Person Households with 2 Autos
HHS3A03	Statewide Percentage of 3 Person Households with 3+ Autos
HHS4A00	Statewide Percentage of 4+ Person Households with 0 Autos
HHS4A01	Statewide Percentage of 4+ Person Households with 1 Autos
HHS4A02	Statewide Percentage of 4+ Person Households with 2 Autos
HHS4A03	Statewide Percentage of 4+ Person Households with 3+ Autos

Source: Cambridge Systematics, Inc.



**Table A.5 Field Descriptions - Home-Based Work Trip Generation Rate Database File:**  
***hbw\_prate.dbf***

<b>Field Name</b>	<b>Field Description</b>
SZ	Superzone
HH1A00	Trip Rate for 1 Person Household – 0 Autos
HH1A01	Trip Rate for 1 Person Household – 1 Autos
HH1A02	Trip Rate for 1 Person Household – 2 Autos
HH1A03	Trip Rate for 1 Person Household – 3+ Autos
HH2A00	Trip Rate for 2 Person Household – 0 Autos
HH2A01	Trip Rate for 2 Person Household – 1 Autos
HH2A02	Trip Rate for 2 Person Household – 2 Autos
HH2A03	Trip Rate for 2 Person Household – 3+ Autos
HH3A00	Trip Rate for 3 Person Household – 0 Autos
HH3A01	Trip Rate for 3 Person Household – 1 Autos
HH3A02	Trip Rate for 3 Person Household – 2 Autos
HH3A03	Trip Rate for 3 Person Household – 3+ Autos
HH4A00	Trip Rate for 4+ Person Household – 0 Autos
HH4A01	Trip Rate for 4+ Person Household – 1 Autos
HH4A02	Trip Rate for 4+ Person Household – 2 Autos
HH4A03	Trip Rate for 4+ Person Household – 3+ Autos

**Source:** Cambridge Systematics, Inc.



**Table A.6 Field Descriptions - Home-Based Other Trip Generation Rate Database File: *hbo\_prate.dbf***

<b>Field Name</b>	<b>Field Description</b>
SZ	Superzone
HH1A00	Trip Rate for 1 Person Household – 0 Autos
HH1A01	Trip Rate for 1 Person Household – 1 Autos
HH1A02	Trip Rate for 1 Person Household – 2 Autos
HH1A03	Trip Rate for 1 Person Household – 3+ Autos
HH2A00	Trip Rate for 2 Person Household – 0 Autos
HH2A01	Trip Rate for 2 Person Household – 1 Autos
HH2A02	Trip Rate for 2 Person Household – 2 Autos
HH2A03	Trip Rate for 2 Person Household – 3+ Autos
HH3A00	Trip Rate for 3 Person Household – 0 Autos
HH3A01	Trip Rate for 3 Person Household – 1 Autos
HH3A02	Trip Rate for 3 Person Household – 2 Autos
HH3A03	Trip Rate for 3 Person Household – 3+ Autos
HH4A00	Trip Rate for 4+ Person Household – 0 Autos
HH4A01	Trip Rate for 4+ Person Household – 1 Autos
HH4A02	Trip Rate for 4+ Person Household – 2 Autos
HH4A03	Trip Rate for 4+ Person Household – 3+ Autos

**Source:** Cambridge Systematics, Inc.



**Table A.7 Field Descriptions – Non-Home Based Trip Generation Rate Database File: *nhb\_prate.dbf***

<b>Field Name</b>	<b>Field Description</b>
SZ	Superzone
HH1A00	Trip Rate for 1 Person Household – 0 Autos
HH1A01	Trip Rate for 1 Person Household – 1 Autos
HH1A02	Trip Rate for 1 Person Household – 2 Autos
HH1A03	Trip Rate for 1 Person Household – 3+ Autos
HH2A00	Trip Rate for 2 Person Household – 0 Autos
HH2A01	Trip Rate for 2 Person Household – 1 Autos
HH2A02	Trip Rate for 2 Person Household – 2 Autos
HH2A03	Trip Rate for 2 Person Household – 3+ Autos
HH3A00	Trip Rate for 3 Person Household – 0 Autos
HH3A01	Trip Rate for 3 Person Household – 1 Autos
HH3A02	Trip Rate for 3 Person Household – 2 Autos
HH3A03	Trip Rate for 3 Person Household – 3+ Autos
HH4A00	Trip Rate for 4+ Person Household – 0 Autos
HH4A01	Trip Rate for 4+ Person Household – 1 Autos
HH4A02	Trip Rate for 4+ Person Household – 2 Autos
HH4A03	Trip Rate for 4+ Person Household – 3+ Autos

**Source:** Cambridge Systematics, Inc.



**Table A.8 Field Descriptions – Long Trip Purpose Trip Generation Rate Database File: *lng\_prate.dbf***

<b>Field Name</b>	<b>Field Description</b>
SZ	Superzone
HH1A00	Trip Rate for 1 Person Household – 0 Autos
HH1A01	Trip Rate for 1 Person Household – 1 Autos
HH1A02	Trip Rate for 1 Person Household – 2 Autos
HH1A03	Trip Rate for 1 Person Household – 3+ Autos
HH2A00	Trip Rate for 2 Person Household – 0 Autos
HH2A01	Trip Rate for 2 Person Household – 1 Autos
HH2A02	Trip Rate for 2 Person Household – 2 Autos
HH2A03	Trip Rate for 2 Person Household – 3+ Autos
HH3A00	Trip Rate for 3 Person Household – 0 Autos
HH3A01	Trip Rate for 3 Person Household – 1 Autos
HH3A02	Trip Rate for 3 Person Household – 2 Autos
HH3A03	Trip Rate for 3 Person Household – 3+ Autos
HH4A00	Trip Rate for 4+ Person Household – 0 Autos
HH4A01	Trip Rate for 4+ Person Household – 1 Autos
HH4A02	Trip Rate for 4+ Person Household – 2 Autos
HH4A03	Trip Rate for 4+ Person Household – 3+ Autos

**Source:** Cambridge Systematics, Inc.

**Table A.9 Field Descriptions – Internal Indiana Trip Rate Percentages Database File: *intpcts.dbf***

<b>Field Name</b>	<b>Field Description</b>
SZ	Superzone Number
HBWPCT	Home-Based Work Purpose Internal Indiana Percentage
HBOPCT	Home-Based Other Purpose Internal Indiana Percentage
NHBPCT	Non-Home Based Purpose Internal Indiana Percentage
LNGPCT	Long Trip Purpose Internal Indiana Percentage

**Source:** Cambridge Systematics, Inc.



**Table A.10 Field Descriptions – Friction Factor Lookup Table**  
***FFACTORS.bin***

**Database File:**

Field Name	Field Description
Bins	Impedance Value
HBW	Home-Based Work Friction Factor
HBO	Home-Based Other Friction Factor
NHB	Non-Home Based Friction Factor
LNG	Long Trip Purpose Friction Factor

Source: Cambridge Systematics, Inc.

**Table A.11 Field Descriptions – Binary MNL Input File**  
***a dictionary is included when the file is created.***

**Database File: *Ingmodel.bin,***

Field Name	Field Description
Alternative	Alternatives to be analyzed, includes coefficients
Transit_ASC	Alternative Specific Constants for Transit.
Time	Time matrix and matrix core name to use for the alternative
Cost	Cost matrix and matrix core name to use for the alternative
OV_TIME	Out of Vehicle time matrix and matrix core to use for the alternative

Source: Cambridge Systematics, Inc.



# **Appendix B**

## *1998 Updated Indiana Model GIS DK Program*



```

Dbox "startdbx",,50 title: "4 Step Model Process"  ToolBox
  init do
    runmacro("closematrices",)
    layer_names = GetLayerNames()
    theviews = GetViewNames()
    todlist = {"AM Peak","Final AM Peak"}, {"PM Peak","Final PM Peak"}, {"OFF Peak","Final
OFF Peak"}}
  enditem
// Create the Left Side of the Dialog Box First

Frame 1,0.5,23,19 Prompt: "General Settings"
Text "Line Layer" 2.5, 1.75
Popdown Menu "Lines" 2.5, 3
  List: layer_names
  variable: linevwidx
  do
    linevw = layer_names[linevwidx]
    lineinfo = GetLayerInfo(linevw)
    linefilename = lineinfo[10]
  enditem

Text "Node Layer" 2.5, 4.75
Popdown Menu "Nodes" 2.5, 6
  List: layer_names
  variable: nodevwidx
  do
    nodevw = layer_names[nodevwidx]
    nodeinfo = GetLayerInfo(nodevw)
    nodefilename = nodeinfo[10]
  enditem

text "TAZ Layer" 2.5,7.75
Popdown Menu "Taz" 2.5, 9
  List: layer_names
  variable: tazvwidx
  do
    tazvw = layer_names[tazvwidx]
    tazinfo = GetLayerInfo(tazvw)
    tazfilename = tazinfo[10]
  enditem

text "Indiana Zones" 2.5,10.75
Edit Int "internals" 2.5, 12
  variable: intzones

text "External Area" 2.5,13.75
Edit Int "internals" 2.5, 15
  variable: extzones

text "Total Zones" 2.5,16.75
Edit Int "allzones" 2.5, 18
  variable: allzones

// Now create the right side of the Dialog Box

Frame 25, 0.5 , 24, 16 Prompt: "Modeling Modules"

```



```
button "Trip Generation" 27,2,20 do
  hhdist = runMacro("create_hhdist",{theviews,thepath,tazvw,extzones})
enditem

button "Trip Distribution" 27,4.5,20 do
  RunMacro("balance_ps_to_as",{thepath,tazvw,tazfilename})
  zonelist = {intzones,extzones}
  for a = 1 to 2 do
    zones = zonelist[a]
    cnet = runMacro("create_hnet",{thepath,linevw,nodevw,zones,a})
    impmat =
RunMacro("Multi_Path",{thepath,nodevw,tazvw,linefilename,zones,a,cnet})
    RunMacro("friction_factors",{thepath,impmat,a})
    RunMacro("TripDist",{thepath,tazvw,tazfilename,a,intzones})
  end
enditem

button "Mode Choice" 27,7,20 do
  RunMacro("thematrix",thepath)
  RunMacro("mnl_eval_lng",{thepath,tazvw,tazfilename,intzones})
  runMacro("test_matrix",thepath)
enditem

button "Time of Day" 27, 9.5,20 do
  runMacro("matrix_manip",thepath)
enditem

button "Traffic Assignment" 27, 12, 20 do
  for t = 1 to todlist.length do
    trucknet = RunMacro("create_hnet",{thepath,linevw,nodevw,allzones,2})
    ajvw = runmacro("truckassign",{thepath,linefilename,linevw,trucknet,todlist[t][1]})
    finalnet = RunMacro("create_hnet",{thepath,ajvw,nodevw,allzones,3})
    closeview(ajvw)
    runmacro("assign",{thepath,linefilename,linevw,finalnet,todlist[t][2]})
  end
  runmacro("finalcalcs",{linevw})
enditem

button "Run All Steps" 27,14.5,20 do
  hhdist = runMacro("create_hhdist",{theviews,thepath,tazvw,extzones})
  RunMacro("balance_ps_to_as",{thepath,tazvw,tazfilename})
  zonelist = {intzones,extzones}
  for a = 1 to 2 do
    zones = zonelist[a]
    cnet = runMacro("create_hnet",{thepath,linevw,nodevw,zones,a})
    impmat =
RunMacro("Multi_Path",{thepath,nodevw,tazvw,linefilename,zones,a,cnet})
    RunMacro("friction_factors",{thepath,impmat,a})
    RunMacro("TripDist",{thepath,tazvw,tazfilename,a,intzones})
  end
  RunMacro("thematrix",thepath)
  RunMacro("mnl_eval_lng",{thepath,tazvw,tazfilename,intzones})
  RunMacro("test_matrix",thepath)
  runMacro("matrix_manip",thepath)
```

```

    for t = 1 to todlist.length do
        trucknet = RunMacro("create_hnet",{thepath,linevw,nodevw,allzones,2})
        ajvw = runmacro("truckassign",{thepath,linefilename,linevw,trucknet,todlist[t][1]})
        finalnet = RunMacro("create_hnet",{thepath,ajvw,nodevw,allzones,3})
        closeview(ajvw)
        runmacro("assign",{thepath,linefilename,linevw,finalnet,todlist[t][2]})
    end
    runmacro("finalcalcs",{linevw})
enditem

// Now create the Path part of the Dialog box on the bottom

Frame 1, 20 , 48, 3 Prompt: "Modeling Path"
button "Browse" 2.5,21.5, 10 do
    thepath = ChooseDirectory("Choose the Modeling Directory")
enditem

text "Model Path" 15.5,21.5,32 Framed variable: thepath

Frame 1,23,48,2.5
text "Alternative Code" 10.5,24,15
Edit Text "Alternative" 26,24,10
    variable: alt

// Now put the cancel button on the bottom of the dialog box so we can get out of it

button "Cancel" 17.5, 27, 15 do return() enditem
endDbox

Dbox "Enter_Value" (in_value) Title: in_value[2]

    //Text 1,1
    Edit Text "num iter item" 1,1,30
    //prompt: "Please enter path of Model Files"
    Variable: thepath
    do
        Return(thepath)
    enditem
endDbox

Dbox "selectview" (in_value) Title: in_value[1]
    init do
        if in_value[2].length = 0 then do
            ShowMessage("There are no datafiles loaded")
            return()
            field_idx = 1
        end
    enditem

    scroll list 2.5,1,25,10
    list: in_value[2]
    Variables: field_idx

    button "OK" 30, 1, 9 do
        Return(field_idx)
    enditem
    button "Cancel"30, 3, 9 do
        return ()
    enditem

```



```
        enditem
endDbox

//***** This Macro Creates the Highway Network *****

Macro "create_hnet" (in_value)
  thepath = in_value[1]
  vw = in_value[2]
  nodes = in_value[3]
  zones = string(in_value[4])
  flag = in_value[5]
  setview(vw)
  if flag = 1 then do
    nettype = "int"
    thenetwork = CreateNetwork(null, thepath+"\\hwy\\highway_"+nettype+".net", "Highway
Network",

    {{ "FFTIME", "FFTIME" }, {"Length", "Length"}, {"CTIME", "CTIME"}, {"AB_CAPPEAK", "AB_CAPPEAK"}, {"BA
_CAPPEAK", "BA_CAPPEAK"},
    {"AB_CAP", "AB_CAP"}, {"BA_CAP", "BA_CAP"}}, , , )
    end
  if flag = 2 then do
    nettype = "intext"
    thenetwork = CreateNetwork(null, thepath+"\\hwy\\highway_"+nettype+".net", "Highway
Network",

    {{ "FFTIME", "FFTIME" }, {"Length", "Length"}, {"CTIME", "CTIME"}, {"AB_CAPPEAK", "AB_CAPPEAK"}, {"BA
_CAPPEAK", "BA_CAPPEAK"},
    {"AB_CAP", "AB_CAP"}, {"BA_CAP", "BA_CAP"}}, , , )
    end
  if flag = 3 then do
    nettype = "final"
    thenetwork = CreateNetwork(null, thepath+"\\hwy\\highway_"+nettype+".net", "Highway
Network",

    {{ "FFTIME", "FFTIME" }, {"Length", "Length"}, {"CTIME", "CTIME"}, {"AB_CAPPEAK", "AB_CAPPEAK"}, {"BA
_CAPPEAK", "BA_CAPPEAK"},
    {"AB_CAP", "AB_CAP"}, {"BA_CAP", "BA_CAP"}, {"AB_FLOW", "AB_FLOW"}, {"BA_FLOW", "BA_FLOW"}}, , , )
    end
  nodelayer = nodes
  SetLayer(nodelayer)
  nsetname = "taz"
  qry = "Select * where taz <="+zones
  n = SelectByQuery(nsetname, "Several", qry,)
  SetCursor("Hourglass")
  idxSet = nodelayer + "|" + nsetname
  opts = {{ "Use Centroids", "True"},
    {"Use Turn Penalties", "False"},
    {"Centroids Set", idxSet}}

  ChangeNetworkSettings(thenetwork, opts)
  netinfo = GetNetworkInfo(thenetwork)
  netfile = netinfo[1]
```

```

return(netfile)
endMacro

//***** END Create Highway Macro
*****

// *****This Is the Trip Generation Section
*****

// *****This macro creates the Household Distribution
*****

macro "create_hhdist" (in_value)
  theviews = in_value[1]
  thepath = in_value[2]
  tzvw = in_value[3]
  intzone = in_value[4]

  distvw = runmacro("create_table",{temp_hh_dist,"temp_hh.dbf","hhdist",thepath+"\\tg"})
  pctvw = opentable("percents","DBASE",{thepath+"\\tg\\pcts_exp.dbf",})
  stwpctvw = opentable("state_pcts","DBASE",{thepath+"\\tg\\swtpct.dbf",})
  pcttavvw = JoinViews("TAZ+PCTS",tzvw+".TAZ_ID",pctvw+".TAZ_ID1",)
  //thequery = "Select * where ID <= "+intzones
  //setview(pcttavvw)
  //SelectByQuery("TAZS","Several",thequery,)
  tzrec = GetFirstRecord(pcttavvw+"|",null)
  // declare variables

  totalhh = 0
  ttlhh1ao0 = 0
  ttlhh1ao1 = 0
  ttlhh1ao2 = 0
  ttlhh1ao3 = 0
  ttlhh2ao0 = 0
  ttlhh2ao1 = 0
  ttlhh2ao2 = 0
  ttlhh2ao3 = 0
  ttlhh3ao0 = 0
  ttlhh3ao1 = 0
  ttlhh3ao2 = 0
  ttlhh3ao3 = 0
  ttlhh4ao0 = 0
  ttlhh4ao1 = 0
  ttlhh4ao2 = 0
  ttlhh4ao3 = 0

  while tzrec <> null do
    tazid = pcttavvw.("TAZ_ID")
    sz = pcttavvw.("SZ")
    if sz <> null then do
      hholds = pcttavvw.("HOUSEHOLDS")
      totalhh = hholds+totalhh
      pcthh1 = pcttavvw.("PCTHHS1")
      pcthh2 = pcttavvw.("PCTHHS2")
      pcthh3 = pcttavvw.("PCTHHS3")
      pcthh4 = pcttavvw.("PCTHHS4")
    enddo
  endwhile
endmacro

```



```
pctAO0 = pttazvw.("PCTAO0")
pctAO1 = pttazvw.("PCTAO1")
pctAO2 = pttazvw.("PCTAO2")
pctAO3 = pttazvw.("PCTAO3")
hh1ao0 = hholds*pcthh1*pctAO0
hh1ao1 = hholds*pcthh1*pctAO1
hh1ao2 = hholds*pcthh1*pctAO2
hh1ao3 = hholds*pcthh1*pctAO3
hh2ao0 = hholds*pcthh2*pctAO0
hh2ao1 = hholds*pcthh2*pctAO1
hh2ao2 = hholds*pcthh2*pctAO2
hh2ao3 = hholds*pcthh2*pctAO3
hh3ao0 = hholds*pcthh3*pctAO0
hh3ao1 = hholds*pcthh3*pctAO1
hh3ao2 = hholds*pcthh3*pctAO2
hh3ao3 = hholds*pcthh3*pctAO3
hh4ao0 = hholds*pcthh4*pctAO0
hh4ao1 = hholds*pcthh4*pctAO1
hh4ao2 = hholds*pcthh4*pctAO2
hh4ao3 = hholds*pcthh4*pctAO3
//if hh1ao0 = null then do
```

```
//showarray({tazid,sz,totalhh,pcthh1,pcthh2,pcthh3,pcthh4,pctAO0,pctAO1,pctAO2,pctAO3,hh1ao0,hh1ao1,hh1ao2,
```

```
hh1ao3,hh2ao0,hh2ao1,hh2ao2,hh2ao3,hh3ao0,hh3ao1,hh3ao2,hh3ao3,hh4ao0,hh4ao1,hh4ao2,hh4ao3})
```

```
//end
ttlhh1ao0 = hh1ao0 + ttlhh1ao0
ttlhh1ao1 = hh1ao1 + ttlhh1ao1
ttlhh1ao2 = hh1ao2 + ttlhh1ao2
ttlhh1ao3 = hh1ao3 + ttlhh1ao3
ttlhh2ao0 = hh2ao0 + ttlhh2ao0
ttlhh2ao1 = hh2ao1 + ttlhh2ao1
ttlhh2ao2 = hh2ao2 + ttlhh2ao2
ttlhh2ao3 = hh2ao3 + ttlhh2ao3
ttlhh3ao0 = hh3ao0 + ttlhh3ao0
ttlhh3ao1 = hh3ao1 + ttlhh3ao1
ttlhh3ao2 = hh3ao2 + ttlhh3ao2
ttlhh3ao3 = hh3ao3 + ttlhh3ao3
ttlhh4ao0 = hh4ao0 + ttlhh4ao0
ttlhh4ao1 = hh4ao1 + ttlhh4ao1
ttlhh4ao2 = hh4ao2 + ttlhh4ao2
ttlhh4ao3 = hh4ao3 + ttlhh4ao3
addrrecords(distvw,
```

```
{ "TAZID", "SZ", "HHS1AO0", "HHS1AO1", "HHS1AO2", "HHS1AO3", "HHS2AO0", "HHS2AO1", "HHS2AO2",
  "HHS2AO3", "HHS3AO0", "HHS3AO1", "HHS3AO2", "HHS3AO3", "HHS4AO0", "HHS4AO1",
```

```
"HHS4AO2", "HHS4AO3"}, {{tazid,sz,hh1ao0,hh1ao1,hh1ao2,hh1ao3,hh2ao0,hh2ao1,hh2ao2,
  hh2ao3,hh3ao0,hh3ao1,hh3ao2,hh3ao3,hh4ao0,hh4ao1,hh4ao2,hh4ao3}}, null)
```

```
end
```

```
tzrec = GetNextRecord(pcttazvw+"|", null, null)
```

```
end
```

```

//showarray({ttlhh1ao0,ttlhh1ao1,ttlhh1ao2,ttlhh1ao3,ttlhh2ao0,ttlhh2ao1,ttlhh2ao2,ttlhh2ao
2,ttlhh3ao0,ttlhh3ao1,
//          ttlhh3ao2,ttlhh3ao3,ttlhh4ao0,ttlhh4ao1,ttlhh4ao2,ttlhh4ao3})

closeview(pcttavw)
finaldistvw =
runmacro("create_table",{ "final_hh_dist", "final_hh.dbf", "hhdist", thepath+"\\tg"})

// do adjustment to statewide percentages here

swprec = GetFirstRecord(stwpctvw+"|",null)
while swprec <> null do
  swhh1ao0 = stwpctvw.("HHS1AO0")
  swhh1ao1 = stwpctvw.("HHS1AO1")
  swhh1ao2 = stwpctvw.("HHS1AO2")
  swhh1ao3 = stwpctvw.("HHS1AO3")
  swhh2ao0 = stwpctvw.("HHS2AO0")
  swhh2ao1 = stwpctvw.("HHS2AO1")
  swhh2ao2 = stwpctvw.("HHS2AO2")
  swhh2ao3 = stwpctvw.("HHS2AO3")
  swhh3ao0 = stwpctvw.("HHS3AO0")
  swhh3ao1 = stwpctvw.("HHS3AO1")
  swhh3ao2 = stwpctvw.("HHS3AO2")
  swhh3ao3 = stwpctvw.("HHS3AO3")
  swhh4ao0 = stwpctvw.("HHS4AO0")
  swhh4ao1 = stwpctvw.("HHS4AO1")
  swhh4ao2 = stwpctvw.("HHS4AO2")
  swhh4ao3 = stwpctvw.("HHS4AO3")
swprec = GetNextRecord(stwpctvw+"|",null,null)
end

adjfhh1ao0 = (swhh1ao0*totalhh) / ttlhh1ao0
adjfhh1ao1 = (swhh1ao1*totalhh) / ttlhh1ao1
adjfhh1ao2 = (swhh1ao2*totalhh) / ttlhh1ao2
adjfhh1ao3 = (swhh1ao3*totalhh) / ttlhh1ao3
adjfhh2ao0 = (swhh2ao0*totalhh) / ttlhh2ao0
adjfhh2ao1 = (swhh2ao1*totalhh) / ttlhh2ao1
adjfhh2ao2 = (swhh2ao2*totalhh) / ttlhh2ao2
adjfhh2ao3 = (swhh2ao3*totalhh) / ttlhh2ao3
adjfhh3ao0 = (swhh3ao0*totalhh) / ttlhh3ao0
adjfhh3ao1 = (swhh3ao1*totalhh) / ttlhh3ao1
adjfhh3ao2 = (swhh3ao2*totalhh) / ttlhh3ao2
adjfhh3ao3 = (swhh3ao3*totalhh) / ttlhh3ao3
adjfhh4ao0 = (swhh4ao0*totalhh) / ttlhh4ao0
adjfhh4ao1 = (swhh4ao1*totalhh) / ttlhh4ao1
adjfhh4ao2 = (swhh4ao2*totalhh) / ttlhh4ao2
adjfhh4ao3 = (swhh4ao3*totalhh) / ttlhh4ao3

tmpdistrec = GetFirstRecord(distvw+"|",null)

while tmpdistrec <> null do
  tazid2 = distvw.("TAZID")
  sz2 = distvw.("SZ")
  temphh1a0 = distvw.("HHS1AO0")
  adjhh1a0 = temphh1a0* adjfhh1a0
  temphh1a1 = distvw.("HHS1AO1")
  adjhh1a1 = temphh1a1 * adjfhh1a1
  temphh1a2 = distvw.("HHS1AO2")
  adjhh1a2 = temphh1a2 * adjfhh1a2

```



```
temphh1a3 = distvw("HHS1AO3")
adjhh1ao3 = temphh1a3 * adjfhh1ao3
temphh2a0 = distvw("HHS2AO0")
adjhh2ao0 = temphh2a0 * adjfhh2ao0
temphh2a1 = distvw("HHS2AO1")
adjhh2ao1 = temphh2a1 * adjfhh2ao1
temphh2a2 = distvw("HHS2AO2")
adjhh2ao2 = temphh2a2 * adjfhh2ao2
temphh2a3 = distvw("HHS2AO3")
adjhh2ao3 = temphh2a3 * adjfhh2ao3
temphh3a0 = distvw("HHS3AO0")
adjhh3ao0 = temphh3a0 * adjfhh3ao0
temphh3a1 = distvw("HHS3AO1")
adjhh3ao1 = temphh3a1 * adjfhh3ao1
temphh3a2 = distvw("HHS3AO2")
adjhh3ao2 = temphh3a2 * adjfhh3ao2
temphh3a3 = distvw("HHS3AO3")
adjhh3ao3 = temphh3a3 * adjfhh3ao3
temphh4a0 = distvw("HHS4AO0")
adjhh4ao0 = temphh4a0 * adjfhh4ao0
temphh4a1 = distvw("HHS4AO1")
adjhh4ao1 = temphh4a1 * adjfhh4ao1
temphh4a2 = distvw("HHS4AO2")
adjhh4ao2 = temphh4a2 * adjfhh4ao2
temphh4a3 = distvw("HHS4AO3")
adjhh4ao3 = temphh4a3 * adjfhh4ao3
addrecords(finaldistvw,
{"TAZID", "SZ", "HHS1AO0", "HHS1AO1", "HHS1AO2", "HHS1AO3", "HHS2AO0", "HHS2AO1", "HHS2AO2",
"HHS2AO3", "HHS3AO0", "HHS3AO1", "HHS3AO2", "HHS3AO3", "HHS4AO0", "HHS4AO1",
```

```
"HHS4AO2", "HHS4AO3"}, {tazid2, sz2, adjhh1ao0, adjhh1ao1, adjhh1ao2, adjhh1ao3, adjhh2ao0, adjhh2ao1, adjhh2ao2,
```

```
adjhh2ao3, adjhh3ao0, adjhh3ao1, adjhh3ao2, adjhh3ao3, adjhh4ao0, adjhh4ao1, adjhh4ao2, adjhh4ao3}}, null)
tmpdistrec = GetNextRecord(distvw+"|", null, null)
```

```
end
closeview(distvw)
//createeditor("final_hh_dist", finaldistvw+"|", null,)
deletefile(thepath+"\\tg\\temp_hh.dbf")
deletefile(thepath+"\\tg\\temp_hh.mdx")
prdvw = runmacro("create_tripdist", {finaldistvw, theviews, thepath})
attrmodel = runmacro("attraction_model", {tzvw, prdvw, thepath})
endmacro
```

```
// *****End Household distribution Macro
*****
```

```
// *****This section creates the production rates
*****
```

```
macro "create_tripdist" (in_value)
```

```
theviews = in_value[2]
newdistvw = in_value[1]
thepath = in_value[3]
```

```
hbwprd = opentable("hbw_prate", "DBASE", {thepath+"\\tg\\hbwprate_30.dbf", })
```

```

hboprdr = opentable("hbo_prater", "DBASE", {thepath+"\\tg\\hboprater_30.dbf",})
nhboprdr = opentable("nhb_prater", "DBASE", {thepath+"\\tg\\nhboprater_30.dbf",})
lngprdr = opentable("lng_prater", "DBASE", {thepath+"\\tg\\lngoprater_30.dbf",})
trppct = opentable("internal_pcts", "DBASE", {thepath+"\\tg\\intpcts_exp.dbf",})

prdrlist = {hboprdr, nhboprdr, lngprdr}
triplist = {"hbw_trip", "hbo_trip", "nhb_trip", "lng_trip"}
prdrvw = runmacro("create_table", {"productions", "product.dbf", "prod", thepath+"\\tg"})
for i = 1 to prdrlist.length do
    prdrvw = prdrlist[i]
    triptab = triplist[i]
    purp = left(triptab,3)
    tripvw = runmacro("create_table", {triptab, triptab+".dbf", "trip", thepath+"\\tg"})

    prec = GetFirstRecord(prdrvw+"|", null)

    while prec <> null do

        sz3 = prdrvw.("SZ")
        aquery = "Select * where SZ = "+string(sz3)
        setview (trppct)
        account = selectbyquery("trippct", "Several", aquery,)

        if account > 1 then do
            showmessage("Problem with trippct table!")
            return ()
        end

        pctrec = GetFirstRecord(trppct+"|trippct", null)

        while pctrec <> null do
            intpct = trppct.(purp+"pct")
            pctrec = GetNextRecord(trppct+"|trippct", null, null)
        end

        setview(newdistvw)

        SelectByQuery("Tony", "Several", aquery,)
        trec = GetFirstRecord(newdistvw+"|Tony", null)

        while trec <> null do
            tazid3 = newdistvw.("TAZID")
            trphh1ao0 = prdrvw.("HH1AO0") * (newdistvw.("HHS1AO0"))
            trphh1ao1 = prdrvw.("HH1AO1") * (newdistvw.("HHS1AO1"))
            trphh1ao2 = prdrvw.("HH1AO2") * (newdistvw.("HHS1AO2"))
            trphh1ao3 = prdrvw.("HH1AO3") * (newdistvw.("HHS1AO3"))
            trphh2ao0 = prdrvw.("HH2AO0") * (newdistvw.("HHS2AO0"))
            trphh2ao1 = prdrvw.("HH2AO1") * (newdistvw.("HHS2AO1"))
            trphh2ao2 = prdrvw.("HH2AO2") * (newdistvw.("HHS2AO2"))
            trphh2ao3 = prdrvw.("HH2AO3") * (newdistvw.("HHS2AO3"))
            trphh3ao0 = prdrvw.("HH3AO0") * (newdistvw.("HHS3AO0"))
            trphh3ao1 = prdrvw.("HH3AO1") * (newdistvw.("HHS3AO1"))
            trphh3ao2 = prdrvw.("HH3AO2") * (newdistvw.("HHS3AO2"))
            trphh3ao3 = prdrvw.("HH3AO3") * (newdistvw.("HHS3AO3"))
            trphh4ao0 = prdrvw.("HH4AO0") * (newdistvw.("HHS4AO0"))
            trphh4ao1 = prdrvw.("HH4AO1") * (newdistvw.("HHS4AO1"))
            trphh4ao2 = prdrvw.("HH4AO2") * (newdistvw.("HHS4AO2"))
            trphh4ao3 = prdrvw.("HH4AO3") * (newdistvw.("HHS4AO3"))
        end
    end
end

```



```
totaltrps =
trphh1ao0+trphh1ao1+trphh1ao2+trphh1ao3+trphh2ao0+trphh2ao1+trphh2ao2+trphh2ao3
+trphh3ao0+trphh3ao1+trphh3ao2+trphh3ao3+trphh4ao0+trphh4ao1+trphh4ao2+trphh4ao3

totalinttrips = totaltrps * intpct
TazQuery = "Select * where TAZID = "+string(tazid3)

setview(prodvw)
tzcount = selectbyquery("prods","Several",TazQuery,)

if tzcount = 1 then do
    rechan = Getfirstrecord(prodvw+"|prods",null)
    setrecordvalues(prodvw,rechan,{{purp,totalinttrips}})
end
else
    addrecords(prodvw, {"TAZID",purp},{{tazid3,totalinttrips}},null)

addrecords(tripvw,
{"TAZID","SZ","HHS1AO0","HHS1AO1","HHS1AO2","HHS1AO3","HHS2AO0","HHS2AO1","HHS2AO2",
"HHS2AO3","HHS3AO0","HHS3AO1","HHS3AO2","HHS3AO3","HHS4AO0","HHS4AO1",
"HHS4AO2","HHS4AO3","Total","Int_Total"},{{tazid3,sz3,trphh1ao0,trphh1ao1,trphh1ao2,trphh1ao3,trp
hh2ao0,trphh2ao1,trphh2ao2,
trphh2ao3,trphh3ao0,trphh3ao1,trphh3ao2,trphh3ao3,trphh4ao0,trphh4ao1,trphh4ao2,trphh4ao3,totaltr
ps,totalinttrips}},null)
    trec = GetNextRecord(newdistvw+"|Tony",null,null)
    end
    prec = GetNextRecord(prdvw+"|",null,null)
end
closeview(tripvw)
end
closeview(newdistvw)
return(prodvw)
endMacro

// *****End Productions
*****

// ***** This sections does the attraction model
*****

macro "attraction_model" (in_value)
    tzvw = in_value[1]
    prodvw = in_value[2]
    thepath = in_value[3]

    tempattvw =
runmacro("create_table",{"tmpattract","tmpattract.dbf","attract",thepath+"\\tg"})
    tazrec = GetFirstRecord(tzvw+"|",null)

    ttlhb watt = 0
    ttlhboatt = 0
```

```

ttlhbatt = 0
ttlngatt = 0

while tazrec <> null do
    ataz = tzvw.("TAZ_ID")
    hbwatt = (tzvw.("ZFARMIND") * 1.650857) + (2.26989 * tzvw.("ZGOVT")) + (0.59889 *
tzvw.("ZNONIND"))
    hboatt = (tzvw.("HOUSEHOLDS") * 4.878835) + (0.391102 * tzvw.("ZFARMIND")) +
(0.197753 * tzvw.("ZNONINDG"))
    nhbatt = (tzvw.("ZFARMIND") * 1.375648) + (1.727858 * tzvw.("ZNONINDG"))
    lngatt = (tzvw.("ZNUMEMP") * 0.407612)
    if hbwatt <> null then do
        ttlhbwatt = ttlhbwatt + hbwatt
        ttlhboatt = ttlhboatt + hboatt
        ttlnhbatt = ttlnhbatt + nhbatt
        ttlngatt = ttlngatt + lngatt
    end

    addrecords(tempattvw, {"TAZID", "HBW", "NHB", "HBO", "LNG"}, {{ataz, hbwatt, nhbatt, hboatt, lngatt}}
, null)
        tazrec = GetNextRecord(tzvw+"|", null, null)
    end

    prdrec = GetFirstRecord(prodvw+"|", null)
    ttlhbwprd = 0
    ttlhbopr = 0
    ttlnhbprd = 0
    ttlngprd = 0

    while prdrec <> null do

        hbwprd = prodvw.("HBW")
        hbopr = prodvw.("HBO")
        nhbprd = prodvw.("NHB")
        lngprd = prodvw.("LNG")
        ttlhbwprd = hbwprd + ttlhbwprd
        ttlhbopr = hbopr + ttlhbopr
        ttlnhbprd = nhbprd + ttlnhbprd
        ttlngprd = lngprd + ttlngprd

        prdrec = GetNextRecord(prodvw+"|", null, null)
    end

    attvw = runmacro("create_table", {"attract", "attract.dbf", "attract", thepath+"\\tg"})
    attrec = GetFirstRecord(tempattvw+"|", null)

    while attrec <> null do
        ataz2 = tempattvw.("TAZID")
        finalhbwatt = (tempattvw.("HBW") * ttlhbwprd) / ttlhbwatt
        finalhboatt = (tempattvw.("HBO") * ttlhbopr) / ttlhboatt
        finalnhbatt = (tempattvw.("NHB") * ttlnhbprd) / ttlnhbatt
        finallngatt = (tempattvw.("LNG") * ttlngprd) / ttlngatt

        addrecords(attvw, {"TAZID", "HBW", "NHB", "HBO", "LNG"}, {{ataz2, finalhbwatt, finalnhbatt, finalhbo
att, finallngatt}}, null)

        attrec = GetNextRecord(tempattvw+"|", null, null)
    end
end

```



```
closeview(tempattvw)
ajvw = JoinViews("TAZ+Attract",tzvw+".TAZ_ID",attvw+".TAZID",)
arec = getfirstrecord(ajvw+"|",null)
attrecs = GetRecordCount(ajvw,null)
CreateProgressbar("Updating Attractions...", "True")
i=0
while arec <> null do
    i = i+1
    apct = ceil(i / attrecs * 100)
    updateprogressbar("Record #"+String(i),apct)
    ajvw.hbw_attr = ajvw.HBW
    ajvw.hbo_attr = ajvw.HBO
    ajvw.nhb_attr = ajvw.NHB
    ajvw.lng_attr = ajvw.LNG
    arec = GetNextRecord(ajvw+"|",null,null)
end
destroyprogressbar()
closeview(ajvw)

// put productions values on node layer

pjvw = JoinViews("TAZ+Productions",tzvw+".TAZ_ID",prodvw+".TAZID",)
prec = getfirstrecord(pjvw+"|",null)
CreateProgressbar("Updating Productions...", "True")
i=0
while prec <> null do
    i=i+1
    apct = ceil(i / attrecs * 100)
    updateprogressbar("Record #"+String(i),apct)
    pjvw.hbw_prod = pjvw.HBW
    pjvw.hbo_prod = pjvw.HBO
    pjvw.nhb_prod = pjvw.NHB
    pjvw.lng_prod = pjvw.LNG
    prec = GetNextRecord(pjvw+"|",null,null)
end
destroyprogressbar()
closeview(attvw)
closeview(pjvw)
closeview(prodvw)
deletefile(thepath+"\\tg\\tmpattract.dbf")
deletefile(thepath+"\\tg\\tmpattract.mdx")
endMacro

Macro "balance_ps_to_as" (in_value)

thepath = in_value[1]
tazvw = in_value[2]
tazfilename = in_value[3]
trec = getfirstrecord(tazvw+"|",null)
while trec <> null do
    tazvw.nhb_prod = tazvw.nhb_attr
    trec = GetNextRecord(tazvw+"|",null,null)
end
RunMacro("TCB Init")
// STEP 1: Balance
```

```

Opts = {{"Input",      {"Data Set",      {tazfilename+"|"+tazvw,
                                     tazvw,
                                     "Selection",
                                     "Select * where HBW_PROD <> null"}}},
        {"Data View",      tazvw}},
{"Field",      {"Vector 1",      {"["+tazvw+].HBW_PROD",
                                ["["+tazvw+].HBO_PROD",
                                ["["+tazvw+].NHB_PROD"]}},
        {"Vector 2",      {"["+tazvw+].HBW_ATTR",
                                ["["+tazvw+].HBO_ATTR",
                                ["["+tazvw+].NHB_ATTR"]}}}},
{"Global",      {"Pairs",      3},
        {"Holding Method",      {1,
                                1,
                                1}},
        {"Percent Weight",      {50,
                                50,
                                50}},
        {"Sum Weight",      {100,
                                100,
                                100}},
        {"V1 Options",      {1,
                                1,
                                1}},
        {"V1 Holding Set",      {,,}},
        {"V2 Options",      {1,
                                1,
                                1}},
        {"V2 Holding Set",      {,,}}}}

if !RunMacro("TCB Run Procedure", "Balance", Opts) then Return( RunMacro("TCB Closing", 0) )
endMacro

//*****End Attraction Model *****
//*****End Trip Generation Macros *****

//***** This Section Does Trip Distribution *****

Macro "Multi_Path" (in_value)
thepath = in_value[1]
nodes = in_value[2]
tazvw = in_value[3]
linefilename = in_value[4]
intzones = string(in_value[5])
pass = string(in_value[6])
thenetwork = in_value[7]
impfilename = "Imp0"+pass+".mtx"
RunMacro("TCB Init")
// STEP 1: TCSPMAT
  Opts = {{"Input",      {"Origin Set",      {linefilename+"|"+nodes,
                                     nodes,
                                     "Selection",
                                     "Select * where TAZ <= "+intzones}},
        {"Destination Set",      {linefilename+"|"+nodes,
                                nodes,
                                "Selection"}}},

```



```

        {"Via Set",          {linefilename+"|"+nodes,
                           nodes}},
        {"Network",        thenetwork}},
{"Field",      {"Nodes",      "["+nodes+"].ID"}},
{"Global",     {"SP Option",   "Matrix"},
               {"Minimize",    "FFTIME"},
               {"Skim Fields", {"Length",
                                   "All"},
                                   {"CTIME",
                                   "All"}}}},
{"Output",    {"Output Matrix", {"Label",
                                   "Shortest Path"},
                                   {"File Name",
                                   thepath+"\\td\\"+impfilename}}}}}

```

```
if !RunMacro("TCB Run Procedure", "TCSPMAT", Opts) then Return( RunMacro("TCB Closing", 0) )
```

```
setview(tazvw)
```

```

aquery = "Select * where TAZ_ID <= "+intzones
numsel = selectbyquery("Tazs","Several",aquery,)
tazrec = GetFirstRecord(tazvw+"|Tazs",null)
impmat = openmatrix(thepath+"\\td\\"+impfilename,"Auto")
theSet = nodes+"|Selection"
newindex =CreateMatrixIndex("TAZID",impmat,"Both",theSet,"ID","TAZ")

```

```
while tazrec <> null do
```

```

    nodeid = tazvw.ID
    newlen = tazvw.ApproxRadius
    newfftime = tazvw.IntraTT
    // ctime = tazvw.CTime

```

```

lencur = CreateMatrixCurrency(impmat,"Length (Skim)",null,null,)
setmatrixvalue(lencur,string(nodeid),string(nodeid),newlen)
//fillmatrix(lencur,string(nodeid),string(nodeid),{"Copy",newlen},{{"Diagonal",0}})
ffcur = CreateMatrixCurrency(impmat,"FFTIME",null,null,)
setmatrixvalue(ffcur,string(nodeid),string(nodeid),newfftime)

```

```

//fillmatrix(ffcur,{string(nodeid)},{string(nodeid)},{ "Copy",newfftime},{ {"Diagonal",0}})
ctcur = CreateMatrixCurrency(impmat,"CTIME (Skim)",null,null,)
setmatrixvalue(ctcur,string(nodeid),string(nodeid),newfftime)

```

```

//fillMatrix(ctcur,{string(nodeid)},{string(nodeid)},{ "Copy",newfftime},{ {"Diagonal",0}})
tazrec = GetNextrecord(tazvw+"|Tazs",null,null)

```

```

end
return(impmat)

```

```
endMacro
```

```
Macro "friction_factors" (in_value)
```

```

thepath = in_value[1]
amatrix = in_value[2]
pass = string(in_value[3])

```

```

//openmatrix(thepath+"\\td\\imp01.mtx","Auto")
createTablefromMatrix(amatrix,thepath+"\\td\\tempimp0"+pass+".bin","FFB",{ {"complete","yes"}
}}
```

```

avw = opentable("FFactors","FFB",{thepath+"\\td\\ffactors.bin",})
impvw = opentable("TempImp","FFB",{thepath+"\\td\\tempimp0"+pass+".bin",})

rec = getfirstrecord(avw+"|",null)
i = 0
count = 1
while rec <> null do
    i = i+1
    tt = avw.Bins
    ffhbw = avw.HBW
    ffhbo = avw.HBO
    ffnhb = avw.NHB
    fflng = avw.LONG

    if i = 1 then do
        ffarray = {{ffhbw,ffhbo,ffnhb,fflng}}
    end
    if i > 1 then do
        ffarray = InsertArrayElements(ffarray,i,{{ffhbw,ffhbo,ffnhb,fflng}})
    end
    rec = GetNextRecord(avw+"|",null,null)
end
if (pass = "1") then thefields = {"LNG"}
if pass = "2" then thefields = {"HBW","HBO","NHB"}
atest = RunMacro("addfields",{thefields,impvw})
EnableProgressbar("Status",1)
CreateProgressbar("Calculating Friction Factors...", "True")
imprec = getfirstrecord(impvw+"|",null)
account = getrecordcount(impvw,null)
i = 0

while imprec <> null do
    i = i+1
    apct = ceil(i / account * 100)
    bol = updateprogressbar("Record #"+String(i),apct)

    ctime = impvw.[CTIME (SKIM)]
    lowctime = Floor(ctime)

    if lowctime > 300 then do
        lowctime = 300
    end

    if lowctime = 0 then
        ctimesubary = ffarray[1]
    else
        ctimesubary = ffarray[lowctime+1]
    end
    hbwff = ctimesubary[1]

    fftime = impvw.FFTIME
    lowfftime = floor(fftime)

    if fftime > 300 then
        lowfftime = 300
    end

    if fftime = 0 then
        ffsubary = ffarray[1]
    end
end

```



```
else
    ffsubary = ffarray[lowfftime+1]

    hboff = ffsubary[2]
    nhbff = ffsubary[3]
    lngff = ffsubary[4]
    if pass = "1" then setrecordvalues(impvw,null,{{"LNG",lngff}})
    if pass = "2" then
setrecordvalues(impvw,null,{{"HBW",hbwoff},{ "HBO",hboff},{ "NHB",nhbff}})
    imprec = getnextrecord(impvw+"|",null,null)

end

destroyprogressbar()

if pass = "1" then do
    newmatrix = CreateMatrixfromview("Friction
Factors",impvw+"|", "RCINDEX", "[RCINDEX:1]", {"LNG"},
    {{ "File Name", thepath+"\\td\\ffact0"+pass+".mtx"},
    { "Type", "Float"},
    { "Sparse", "No"},
    { "Column Major", "No"},
    { "File Based", "Yes"}})
end

if pass = "2" then do

    newmatrix = CreateMatrixfromview("Friction
Factors",impvw+"|", "RCINDEX", "[RCINDEX:1]", {"HBW", "HBO", "NHB"},
    {{ "File Name", thepath+"\\td\\ffact0"+pass+".mtx"},
    { "Type", "Float"},
    { "Sparse", "No"},
    { "Column Major", "No"},
    { "File Based", "Yes"}})

end

closeview(impvw)
closeview(avw)
deletefile(thepath+"\\td\\tempimp0"+pass+".bin")
closematrix(newmatrix)
endMacro

Macro "TripDist" (in_value)
    thepath = in_value[1]
    tazvw = in_value[2]
    tazfilename = in_value[3]
    pass = string(in_value[4])
    intzones = string(in_value[5])

    if (pass = "1") then do
        RunMacro("TCB Init")
        // STEP 1: Balance
        BalOpts = {{ "Input",          {{ "Data Set",          {tazfilename+"|"+tazvw,
                                                tazvw,
```





```

"RCIndex:1"},
{thepath+"\\td\\ffact0"+pass+".mtx",
"HBO",
"RCIndex",
"RCIndex:1"},
{thepath+"\\td\\ffact0"+pass+".mtx",
"NHB",
"RCIndex",
"RCIndex:1"}},
{"Imp Matrix Currencies",  {{thepath+"\\td\\ffact0"+pass+".mtx",
"HBW",
"RCIndex",
"RCIndex:1"},
{thepath+"\\td\\ffact0"+pass+".mtx",
"HBW",
"RCIndex",
"RCIndex:1"},
{thepath+"\\td\\ffact0"+pass+".mtx",
"HBW",
"RCIndex",
"RCIndex:1"}},
{"KF Matrix Currencies",  {{thepath+"\\td\\kfactor.mtx",
"KFACTOR",
"node",
"node"},
{thepath+"\\td\\kfactor.mtx",
"KFACTOR",
"node",
"node"},
{thepath+"\\td\\kfactor.mtx",
"KFACTOR",
"node",
"node"}},
{"Field",  {{{"Prod Fields",  {"["+tazvw+"].HBW_PROD",
["+tazvw+"].HBO_PROD",
["+tazvw+"].NHB_PROD"}},
{"Attr Fields",  {"["+tazvw+"].HBW_ATTR",
["+tazvw+"].HBO_ATTR",
["+tazvw+"].NHB_ATTR"}},
{"Global",  {{{"Purpose Names",  {"HBW",
"HBO",
"NHB"}},
{"Iterations",  {200,
200,
200}},
{"Convergence",  {0.01,
0.01,
0.01}},
{"Constraint Type",  {"Double",
"Double",
"Double"}},
{"Fric Factor Type",  {"Matrix",
"Matrix",
"Matrix"}},
{"A List",  {1,

```

```

1,
1}},
{"B List",          {0.3,
0.3,
0.3}},
{"C List",          {0.01,
0.01,
0.01}}},
{"Flag",          {"Use K Factors",    {1,
1,
1}}}},
{"Output",        {"Output Matrix",    {"Label",
"Output Matrix",
"File Name",
thepath+"\\td\\ptripdist0"+pass+".mtx"}}}}}
end
RunMacro("TCB Init")
if !RunMacro("TCB Run Procedure", "Gravity", Opts) then Return( RunMacro("TCB Closing", 0) )
Return()
endMacro

```

```

//*****End Trip Distribution Macros *****
//*****No do the mode choice part *****

```

```

Macro "thematrix" (in_value)
thepath = in_value
// open impedance matrix
M = openmatrix(thepath+"\\td\\imp01.mtx", "True")
// do time matrix here

mc = CreateMatrixCurrency(M, "FFTIME", null, null, )

timematrix = copymatrix(mc, {"File Name", thepath+"\\mc\\time.mtx"},
{"Label", "Time"},
{"File Based", "Yes"})
dropmatrixcore(timematrix, "Length (Skim)")
setmatrixcorename(timematrix, "FFTIME", "FFLOW*")
setmatrixcorename(timematrix, "CTIME (Skim)", "CTIME")
addmatrixcore(timematrix, "FFTRAN_IVTT")
addmatrixcore(timematrix, "FFTRAN_OVTT")
addmatrixcore(timematrix, "CTTRAN_IVTT")
addmatrixcore(timematrix, "CTTRAN_OVTT")

tranmtx = openmatrix(thepath+"\\routes\\TR_skim.mtx", "Auto")

ffovttmc = CreateMatrixCurrency(tranmtx, "FFTIME (non-transit)", null, null, )
ctivttmc = CreateMatrixCurrency(tranmtx, "CTIME (in-vehicle)", null, null, )
ctovttmc = CreateMatrixCurrency(tranmtx, "CTIME (non-transit)", null, null, )
ffivttmc = CreateMatrixCurrency(tranmtx, "FFTIME (in-vehicle)", null, null, )

targffivt = CreateMatrixCurrency(timematrix, "FFTRAN_IVTT", null, null, )
MergeMatrixElements(targffivt, {ffivttmc}, null, null, {"Force Missing", "Yes"})
targffovt = CreateMatrixCurrency(timematrix, "FFTRAN_OVTT", null, null, )
MergeMatrixElements(targffovt, {ffovttmc}, null, null, {"Force Missing", "Yes"})
targctivt = CreateMatrixCurrency(timematrix, "CTTRAN_IVTT", null, null, )

```



```

MergeMatrixElements(targetivt,{ctivttmc},null,null,{{"Force Missing","Yes"}})
targetovt = CreateMatrixCurrency(timematrix, "CTTRAN_OVTT",null,null,)
MergeMatrixElements(targetovt,{ctovttmc},null,null,{{"Force Missing","Yes"}})
closematrix(timematrix)

// Now do the cost matrix

costmatrix = copymatrix(mc,{{"File Name",thepath+"\\mc\\cost.mtx"},
    {"Label","Cost"},
    {"File Based","Yes"},
    {"Table","Length (Skim)"}})
addmatrixcore(costmatrix,"Auto")
addmatrixcore(costmatrix,"Transit")
addmatrixcore(costmatrix,"Length(Transit)")

lentranmc =CreateMatrixCurrency(tranmtx,"Length (in-vehicle)",null,null,)
targlenivt = CreateMatrixCurrency(costmatrix, "Length(Transit)",null,null,)
MergeMatrixElements(targlenivt,{lentranmc},null,null,{{"Force Missing","Yes"}})

automc = CreateMatrixCurrency(costmatrix, "Auto",null,null,)
AutoExpr = "30 * [Length (Skim)]"
EvaluateMatrixExpression(automc,AutoExpr,null,null,)
tranmc = CreateMatrixCurrency(costmatrix, "Transit",null,null,)
TranExpr = "19.50 * [Length(Transit)]"
EvaluateMatrixExpression(tranmc,TranExpr,null,null,)
dropmatrixcore(costmatrix,"Length (Skim)")
endMacro

Macro "mnl_eval_lng" (in_value)
    thepath = in_value[1]
    tazvw = in_value[2]
    tazfilename = in_value[3]
    intzones = string(in_value[4])
    RunMacro("TCB Init")
// STEP 1: MNL Evaluation
    Opts = {{"Input",      {"View Set",      {tazfilename+"|"+tazvw,
        tazvw,
        "Selection",
        "Select * where taz <= "+intzones}},
        {"Destination Set", {tazfilename+"|"+tazvw,
        tazvw,
        "Selection",
        "Select * where taz <=
"+intzones}}},
        {"Model Table",      {thepath+"\\MC\\LNGMODEL.BIN"}},
        {"Matrix Currencies", {{thepath+"\\mc\\time.mtx",
        "FFTRAN_IVTT",
        "RCIndex",
        "RCIndex"},
        {thepath+"\\mc\\cost.mtx",
        "Auto",
        "RCIndex",
        "RCIndex"}}}},
        {"Field",      {"ID Field",      "["+tazvw+"].ID"}},
        {"Global",    {"Number of Modes", 2}},

```

```

        {"Model Name",          "Coefficients"}},
{"Flag",      {"Aggregate",    1},
              {"Delete Case",  0}}},
{"Output",   {"Output Matrix", {"Label",
                                "Output Matrix"},
              {"File Name",
               thepath+"\\mc\\lngshares.mtx"}}}}}

```

```

if !RunMacro("TCB Run Procedure", "MNL Evaluation", Opts) then Return( RunMacro("TCB
Closing", 0) )
endMacro

```

```

Macro "test_matrix" (in_value)
  thepath = in_value
  M = openmatrix(thepath+"\\td\\ptripdist02.mtx", "True")
  mchbw = CreateMatrixCurrency(M, "HBW", null, null, )
  hbwmtx = copymatrix(mchbw, {"File Name", thepath+"\\mc\\hbwbase.mtx"},
                        {"Label", "HBW Base"},
                        {"File Based", "Yes"})
  addmatrixcore(hbwmtx, "Auto")
  addmatrixcore(hbwmtx, "Transit")
  addmatrixcore(hbwmtx, "HBW_VEH_TRP")
  autohbwmc = CreateMatrixCurrency(hbwmtx, "Auto", null, null, )
  hbwexp1 = "[HBW] * 0.99"
  EvaluateMatrixExpression(autohbwmc, hbwexp1, null, null, )
  tranhbwmc = CreateMatrixCurrency(hbwmtx, "Transit", null, null, )
  hbwexp2 = "[HBW] * 0.01"
  EvaluateMatrixExpression(tranhbwmc, hbwexp2, null, null, )
  SetMatrixCore(hbwmtx, "Auto")
  hbwaoaccmc = CreateMatrixCurrency(hbwmtx, "HBW_VEH_TRP", null, null, )
  hbwaoaccexp = "[Auto] / 1.20"
  EvaluateMatrixExpression(hbwaoaccmc, hbwaoaccexp, null, null, )

  dropmatrixcore(hbwmtx, "HBO")
  dropmatrixcore(hbwmtx, "NHB")
  dropmatrixcore(hbwmtx, "HBW")

  mchbo = CreateMatrixCurrency(M, "hbo", null, null, )
  hbomtx = copymatrix(mchbo, {"File Name", thepath+"\\mc\\hbobase.mtx"},
                        {"Label", "HBO Base"},
                        {"File Based", "Yes"})
  addmatrixcore(hbomtx, "Auto")
  addmatrixcore(hbomtx, "Transit")
  addmatrixcore(hbomtx, "HBO_VEH_TRP")
  autohbomc = CreateMatrixCurrency(hbomtx, "Auto", null, null, )
  hboexp1 = "[HBO] * 0.92"
  EvaluateMatrixExpression(autohbomc, hboexp1, null, null, )
  tranhbomc = CreateMatrixCurrency(hbomtx, "Transit", null, null, )
  hboexp2 = "[HBO] * 0.08"
  EvaluateMatrixExpression(tranhbomc, hboexp2, null, null, )
  hboaoaccmc = CreateMatrixCurrency(hbomtx, "HBO_VEH_TRP", null, null, )
  hboaoaccexp = "[Auto] / 2.15"
  EvaluateMatrixExpression(hboaoaccmc, hboaoaccexp, null, null, )
  SetMatrixCore(hbomtx, "Auto")
  dropmatrixcore(hbomtx, "HBO")
  dropmatrixcore(hbomtx, "NHB")
  dropmatrixcore(hbomtx, "HBW")
  mcnhb = CreateMatrixCurrency(M, "nhb", null, null, )

```



```
nhbmtx = copymatrix(mcnhb, {"File Name", thepath+"\\mc\\nhbbase.mtx"},
    {"Label", "NHB Base"},
    {"File Based", "Yes"}})
addmatrixcore(nhbmtx, "Auto")
addmatrixcore(nhbmtx, "Transit")
addmatrixcore(nhbmtx, "NHB_VEH_TRP")
autonhbmc = CreateMatrixCurrency(nhbmtx, "Auto", null, null, )
nhbexpl = "[NHB] * 0.96"
EvaluateMatrixExpression(autonhbmc, nhbexpl, null, null, )

trannhbmc = CreateMatrixCurrency(nhbmtx, "Transit", null, null, )
nhbexp2 = "[NHB] * 0.04"
EvaluateMatrixExpression(trannhbmc, nhbexp2, null, null, )

nhbaoccmc = CreateMatrixCurrency(nhbmtx, "NHB_VEH_TRP", null, null, )
nhbaocccexp = "[Auto] / 1.87"
EvaluateMatrixExpression(nhbaoccmc, nhbaocccexp, null, null, )

SetMatrixCore(nhbmtx, "Auto")
dropmatrixcore(nhbmtx, "NHB")
dropmatrixcore(nhbmtx, "HBO")
dropmatrixcore(nhbmtx, "HBW")

// ***** Do Long Trips here
// *****

lngpermat = openmatrix(thepath+"\\td\\ptripdist01.mtx", "True")
lngshrmtx = OpenMatrix(thepath+"\\mc\\lngshares.mtx", "True")
autolngmc = CreateMatrixCurrency(lngshrmtx, "Auto", null, null, )
lngpermc = CreateMatrixCurrency(lngpermat, "LNG", null, null, )

lngtrpmtx = CombineMatrices({autolngmc, lngpermc}, {"File Name",
thepath+"\\mc\\lngbase.mtx"},
    {"Label", "Long Vehicle Trips"},
    {"File Based", "Yes"},
    {"Operation", "Union"}})

AddMatrixCore(lngtrpmtx, "LNG_VEH_TRP")

lngbsmc = CreateMatrixCurrency(lngtrpmtx, "LNG_VEH_TRP", null, null, )
lngexp = "([Auto] * [LNG]) / 3.06"
EvaluateMatrixExpression(lngbsmc, lngexp, null, null, )

// ***** Combine All Trips Here
// *****

totvehtrpmtx = CombineMatrices({hbwaoccmc, hboaoccmc, nhbaoccmc, lngbsmc}, {"File Name",
thepath+"\\mc\\combvehtrp.mtx"},
    {"Label", "Vehicle Trips"},
    {"File Based", "Yes"},
    {"Operation", "Union"}})

AddMatrixCore(totvehtrpmtx, "Total Vehicles")
```

```

totmc = CreateMatrixCurrency(totvehtrpmtx,"Total Vehicles",null,null,)

RunMacro("TCB Init")
// STEP 1: Matrix Formula Fill
Opts = {"Input",      {"Matrix Currency",      {thepath+"\\mc\\combvehtrp.mtx",
                                                "LNG_VEH_TRP",
                                                "Rows",
                                                "Columns"}}}},
      {"Global",      {"Formula",              "if [LNG_VEH_TRP] = null then 0 else
[LNG_VEH_TRP]"}}}}}

if !RunMacro("TCB Run Operation", "Matrix Formula Fill", Opts) then Return( RunMacro("TCB
Closing", 0))

totexp = "[HBW_VEH_TRP]+[HBO_VEH_TRP]+[NHB_VEH_TRP]+[LNG_VEH_TRP]"

EvaluateMatrixExpression(totmc,totexp,null,null,)

transmtx = TransposeMatrix(totvehtrpmtx,{"File Name",thepath+"\\mc\\transpveh.mtx"},
                              {"Label", "Transposed Vehicles"},
                              {"Type", "Double"},
                              {"Sparse", "No"},
                              {"Column Major", "No"},
                              {"File Based", "Yes"}})

AddMatrixCore(totvehtrpmtx,"Half_Vehicles")
AddMatrixCore(transmtx,"Transp_Half_Vehicles")

halfmc = CreateMatrixCurrency(totvehtrpmtx,"Half_Vehicles",null,null,)
halfexp = "[Total Vehicles] / 2"
EvaluateMatrixExpression(halfmc,halfexp,null,null,)

transhalfmc = CreateMatrixCurrency(transmtx,"Transp_Half_Vehicles",null,null,)
transhalfexp = "[Total Vehicles] / 2"
EvaluateMatrixExpression(transhalfmc,transhalfexp,null,null,)

newvehmtx = CombineMatrices({halfmc, transhalfmc}, {"File Name",
thepath+"\\mc\\vehicles.mtx"},
                              {"Label", "New Matrix"},
                              {"File Based", "Yes"},
                              {"Operation", "Union"}})

AddMatrixCore(newvehmtx,"Final OD")
finalodmc = CreateMatrixCurrency(newvehmtx,"Final OD",null,null,)
finalexp = "[Half_Vehicles] + [Transp_Half_Vehicles]"
EvaluateMatrixExpression(finalodmc,finalexp,null,null,)

//everthing good to here.  Should be able to use merge matrix elements

corr18mat = openmatrix(thepath+"\\mc\\corr18_lng.mtx","True")
corr18mc = CreateMatrixCurrency(corr18mat,"CAR98_LNG",null,null,)

finalmatrix = copymatrix(corr18mc,{"File Name",thepath+"\\mc\\FinalVeh.mtx"},
                              {"Label", "Final Assignments Vehicles"},
                              {"File Based", "Yes"}})

AddMatrixCore(finalmatrix,"Final OD")

```



```
odmc =CreateMatrixCurrency(finalmatrix,"Final OD",null,null,)
alexp = "[CAR98_LNG] * 0"
EvaluateMatrixExpression(odmc,alexp,null,null,)

AddMatrixCore(finalmatrix,"All Vehicles")
fnmc = CreateMatrixCurrency(finalmatrix,"All Vehicles",null,null,)
aexp = "[CAR98_LNG] * 0"
EvaluateMatrixExpression(fnmc,aexp,null,null,)
corr18mc = CreateMatrixCurrency(corr18mat,"CAR98_LNG",null,null,)
MergeMatrixElements(odmc,{finalodmc},null,null,{{"Force Missing","No"}})

finalexp = "[CAR98_LNG] + [Final OD]"
EvaluateMatrixExpression(fnmc,finalexp,null,null,)
// showmessage("Finished Mode Choice")
runmacro("closematrices",)
```

endMacro

Macro "matrix\_manip" (in\_value)

```
thepath = in_value

purplist = {"HBW","HBO","NHB","LNG"}

// Time of day factor multi dimension list --for {HBW,HBO,NHB,LNG {AM - PM - OP}}

factlist1 =
{{0.3317,0.3025,0.3658},{0.2416,0.2773,0.5080},{0.1206,0.2517,0.6277},{0.0699,0.2373,0.6928}}

// In this list the PA and AP factors for all purposes PA always comes first

appalist =
{{0.9568,0.0432,0.0913,0.9087,0.5068,0.4932},{0.8993,0.1007,0.3367,0.6633,0.4076,0.5924},{0.5,0.5,0.5,0.5,0.5},{0.9828,0.0172,0.4652,0.5348,0.4632,0.5368}}

for i = 1 to purplist.length do
  purp = purplist[i]
  flist = factlist1[i]
  appafactlist = appalist[i]

  hbwmat = openmatrix(thepath+"\\mc\\"+purp+"base.mtx","True")
  mc = CreateMatrixCurrency(hbwmat,purp+"_VEH_TRP",null,null,)
  newmtx = copymatrix(mc,{{"File Name",thepath+"\\tod\\"+purp+"fpa.mtx"},
    {"Label",""},
    {"File Based","Yes"}})

  addmatrixcore(newmtx,"PA AM Peak")
  addmatrixcore(newmtx,"PA PM Peak")
  addmatrixcore(newmtx,"PA Off Peak")

  mc1 = CreateMatrixCurrency(newmtx,"PA AM Peak",null,null,)

  expl = "("+purp+"_VEH_TRP)*"+string(flist[1])+")"
```

```

EvaluateMatrixExpression(mcl,expl,null,null,)

mc2 = CreateMatrixCurrency(newmtx,"PA PM Peak",null,null,)

exp2 = "(["+purp+"_VEH_TRP]*"+string(flist[2])+")"
EvaluateMatrixExpression(mc2,exp2,null,null,)

mc3 = CreateMatrixCurrency(newmtx,"PA Off Peak",null,null,)

exp3 = "(["+purp+"_VEH_TRP]*"+string(flist[3])+")"
EvaluateMatrixExpression(mc3,exp3,null,null,)

transmtx = TransposeMatrix(newmtx,{{"File Name",thepath+"\\tod\\"+purp+"fap.mtx"},
    {"Label",purp+" AP"},
    {"Type", "Double"},
    {"Sparse", "No"},
    {"Column Major", "No"},
    {"File Based", "Yes"}})

setmatrixcorename(transmtx,"PA AM Peak", "AP AM Peak")
setmatrixcorename(transmtx,"PA PM Peak", "AP PM Peak")
setmatrixcorename(transmtx,"PA OFF Peak", "AP OFF Peak")

paam = CreateMatrixCurrency(newmtx,"PA AM Peak",null,null,)
apam = CreateMatrixCurrency(transmtx,"AP AM Peak",null,null,)
MatrixCellByCell(paam,apam,{{"File Name", thepath+"\\tod\\"+purp+"famod.mtx"},
    {"Label", "AM OD Matrix"},
    {"Type", "Double"},
    {"Sparse", "No"},
    {"Column Major","No"},
    {"File Based", "Yes"},
    {"Force Missing", "No"},
    {"Operator",3},
    {"Scale Left", appafactlist[1]},
    {"Scale Right",appafactlist[2]}})

papm = CreateMatrixCurrency(newmtx,"PA PM Peak",null,null,)
appm = CreateMatrixCurrency(transmtx,"AP PM Peak",null,null,)
MatrixCellByCell(papm,appm,{{"File Name", thepath+"\\tod\\"+purp+"fpmod.mtx"},
    {"Label", "PM OD Matrix"},
    {"Type", "Double"},
    {"Sparse", "No"},
    {"Column Major","No"},
    {"File Based", "Yes"},
    {"Force Missing", "No"},
    {"Operator",3},
    {"Scale Left",appafactlist[3]},
    {"Scale Right",appafactlist[4]}})

paop = CreateMatrixCurrency(newmtx,"PA OFF Peak",null,null,)
apop = CreateMatrixCurrency(transmtx,"AP OFF Peak",null,null,)
MatrixCellByCell(paop,apop,{{"File Name", thepath+"\\tod\\"+purp+"fopod.mtx"},
    {"Label", "OP OD Matrix"},
    {"Type", "Double"},
    {"Sparse", "No"},
    {"Column Major","No"},
    {"File Based", "Yes"},

```



```
        {"Force Missing", "No"},
        {"Operator",3},
        {"Scale Left", appafactlist[5]},
        {"Scale Right", appafactlist[6]}})
    closematrix(newmtx)

end

// Combine AM Peak Matrices here

hbwammtx = openmatrix(thepath+"\\tod\\hbwfamod.mtx","True")
hboammtx = openmatrix(thepath+"\\tod\\hbofamod.mtx","True")
lngammtx = openmatrix(thepath+"\\tod\\lngfamod.mtx","True")
nhbammtx = openmatrix(thepath+"\\tod\\nhbfamod.mtx","True")

setmatrixcorename(hbwammtx,"Matrix 1", "HBW AM Peak")
setmatrixcorename(hboammtx,"Matrix 1", "HBO AM Peak")
setmatrixcorename(lngammtx,"Matrix 1", "LNG AM Peak")
setmatrixcorename(nhbammtx,"Matrix 1", "NHB AM Peak")

hbwammc = CreateMatrixCurrency(hbwammtx,"HBW AM Peak",null,null,)
hboammc = CreateMatrixCurrency(hboammtx,"HBO AM Peak",null,null,)
lngammc = CreateMatrixCurrency(lngammtx,"LNG AM Peak",null,null,)
nhbammc = CreateMatrixCurrency(nhbammtx,"NHB AM Peak",null,null,)

ammtx = CombineMatrices({hbwammc, hboammc,nhbammc,lngammc}, {"File Name",
thepath+"\\tod\\famod.mtx"},
{"Label", "AM Peak OD"},
{"File Based", "Yes"},
{"Operation", "Union"}})

RunMacro("TCB Init")
// STEP 1: Matrix Formula Fill
    Opts = {"Input",      {"Matrix Currency",      {thepath+"\\tod\\famod.mtx",
                                                    "LNG AM Peak",
                                                    "Rows",
                                                    "Columns"}}},
          {"Global",    {"Formula",              "if [LNG AM Peak] = null then 0 else [LNG AM
Peak]"}}}}}

    if !RunMacro("TCB Run Operation", "Matrix Formula Fill", Opts) then Return( RunMacro("TCB
Closing", 0))

    ammc = CreateMatrixCurrency(ammtx,"HBW AM Peak",null,null,)
    amexp = "[HBW AM Peak]+[HBO AM Peak]+[LNG AM Peak]+[NHB AM Peak]"
    evaluatematrixexpression(ammc,amexp,null,null,)
    setmatrixcorename(ammtx,"HBW AM Peak","AM Peak")
    ammc = CreateMatrixCurrency(ammtx,"AM Peak",null,null,)

    dropmatrixcore(ammtx,"HBO AM Peak")
    dropmatrixcore(ammtx,"NHB AM Peak")
    dropmatrixcore(ammtx,"LNG AM Peak")

    closematrix(hbwammtx)
```

```

closematrix(hboammtx)
closematrix(lngammtx)
closematrix(nhbammtx)

hbwpmmtx = openmatrix(thepath+"\\tod\\hbwfpmod.mtx", "True")
hbopmmtx = openmatrix(thepath+"\\tod\\hbofpmod.mtx", "True")
lngpmmtx = openmatrix(thepath+"\\tod\\lngfpmod.mtx", "True")
nhbpmmtx = openmatrix(thepath+"\\tod\\nhbfpmod.mtx", "True")

setmatrixcorename(hbwpmmtx, "Matrix 1", "HBW PM Peak")
setmatrixcorename(hbopmmtx, "Matrix 1", "HBO PM Peak")
setmatrixcorename(lngpmmtx, "Matrix 1", "LNG PM Peak")
setmatrixcorename(nhbpmmtx, "Matrix 1", "NHB PM Peak")

hbwpmmc = CreateMatrixCurrency(hbwpmmtx, "HBW PM Peak", null, null, )
hbopmmc = CreateMatrixCurrency(hbopmmtx, "HBO PM Peak", null, null, )
lngpmmc = CreateMatrixCurrency(lngpmmtx, "LNG PM Peak", null, null, )
nhbpmmc = CreateMatrixCurrency(nhbpmmtx, "NHB PM Peak", null, null, )

pmmtx = CombineMatrices({hbwpmmc, hbopmmc, nhbpmmc, lngpmmc}, {"File Name",
thepath+"\\tod\\fpmod.mtx"},
{"Label", "PM Peak OD"},
{"File Based", "Yes"},
{"Operation", "Union"}})

RunMacro("TCB Init")
// STEP 1: Matrix Formula Fill
Opts = {"Input", {"Matrix Currency", {thepath+"\\tod\\fpmod.mtx",
"LNG PM Peak",
"Rows",
"Columns"}}},
{"Global", {"Formula", "if [LNG PM Peak] = null then 0 else [LNG PM
Peak]}}}}

if !RunMacro("TCB Run Operation", "Matrix Formula Fill", Opts) then Return( RunMacro("TCB
Closing", 0))

pmmc = CreateMatrixCurrency(pmmtx, "HBW PM Peak", null, null, )
pmexp = "[HBW PM Peak]+[HBO PM Peak]+[LNG PM Peak]+[NHB PM Peak]"
evaluatematrixexpression(pmmc, pmexp, null, null, )
setmatrixcorename(pmmtx, "HBW PM Peak", "PM Peak")
pmmc = CreateMatrixCurrency(pmmtx, "PM Peak", null, null, )

dropmatrixcore(pmmtx, "HBO PM Peak")
dropmatrixcore(pmmtx, "NHB PM Peak")
dropmatrixcore(pmmtx, "LNG PM Peak")

closematrix(hbwpmmtx)
closematrix(hbopmmtx)
closematrix(lngpmmtx)
closematrix(nhbpmmtx)

hbwopmtx = openmatrix(thepath+"\\tod\\hbwfopod.mtx", "True")
hboopmtx = openmatrix(thepath+"\\tod\\hbofopod.mtx", "True")
lngopmtx = openmatrix(thepath+"\\tod\\lngfopod.mtx", "True")
nhbopmtx = openmatrix(thepath+"\\tod\\nhbfopod.mtx", "True")

```



```
setmatrixcorename(hbwopmtx,"Matrix 1", "HBW OFF Peak")
setmatrixcorename(hboopmtx,"Matrix 1", "HBO OFF Peak")
setmatrixcorename(lngopmtx,"Matrix 1", "LNG OFF Peak")
setmatrixcorename(nhbopmtx,"Matrix 1", "NHB OFF Peak")

hbwopmc = CreateMatrixCurrency(hbwopmtx,"HBW OFF Peak",null,null,)
hboopmc = CreateMatrixCurrency(hboopmtx,"HBO OFF Peak",null,null,)
lngopmc = CreateMatrixCurrency(lngopmtx,"LNG OFF Peak",null,null,)
nhbopmc = CreateMatrixCurrency(nhbopmtx,"NHB OFF Peak",null,null,)

opmtx = CombineMatrices({hbwopmc,hboopmc,nhbopmc,lngopmc}, {"File Name",
thepath+"\\tod\\fopod.mtx"},
{"Label", "OFF Peak OD"},
{"File Based", "Yes"},
{"Operation", "Union"}})

RunMacro("TCB Init")
// STEP 1: Matrix Formula Fill
Opts = {"Input", {"Matrix Currency", {thepath+"\\tod\\fopod.mtx",
"LNG OFF Peak",
"Rows",
"Columns"}}},
{"Global", {"Formula", "if [LNG OFF Peak] = null then 0 else [LNG OFF
Peak]}}}]

if !RunMacro("TCB Run Operation", "Matrix Formula Fill", Opts) then Return( RunMacro("TCB
Closing", 0))

opmc = CreateMatrixCurrency(opmtx,"HBW OFF Peak",null,null,)
opexp = "[HBW OFF Peak]+[HBO OFF Peak]+[LNG OFF Peak]+[NHB OFF Peak]"
evaluatematrixexpression(opmc,opexp,null,null,)
setmatrixcorename(opmtx,"HBW OFF Peak","OFF Peak")
opmc = CreateMatrixCurrency(opmtx,"OFF Peak",null,null,)

dropmatrixcore(opmtx,"HBO OFF Peak")
dropmatrixcore(opmtx,"NHB OFF Peak")
dropmatrixcore(opmtx,"LNG OFF Peak")

// Do Corrl8 stuff here

amtx = openmatrix(thepath+"\\mc\\corrl8_lng.mtx","True")
amc = CreateMatrixCurrency(amtx,"CAR98_LNG",null,null,)
corrl8mtx = copymatrix(amc,{"File Name",thepath+"\\tod\\todcorrl8.mtx"},
{"Label","Corridor 18 External"},
{"File Based","Yes"}})

addmatrixcore(corrl8mtx,"Corrl8 AM Peak")
addmatrixcore(corrl8mtx,"Corrl8 PM Peak")
addmatrixcore(corrl8mtx,"Corrl8 OFF Peak")

cammc = CreateMatrixCurrency(corrl8mtx,"Corrl8 AM Peak",null,null,)
camexp = "[CAR98_LNG] * 0.2063"
EvaluateMatrixExpression(cammc,camexp,null,null,)
```

```

cpmmc = CreateMatrixCurrency(corr18mtx,"Corr18 PM Peak",null,null,)
cpmexp = "[CAR98_LNG] * 0.2373"
EvaluateMatrixExpression(cpmmmc,cpmexp,null,null,)

copmc = CreateMatrixCurrency(corr18mtx,"Corr18 OFF Peak",null,null,)
copexp = "[CAR98_LNG] * 0.5564"
EvaluateMatrixExpression(copmc,copexp,null,null,)

aexp = "[CAR98_LNG] * 0"
addmatrixcore(corr18mtx,"AM Peak")
theammc = CreateMatrixCurrency(corr18mtx,"AM Peak",null,null,)
EvaluateMatrixExpression(theammc,aexp,null,null,)
addmatrixcore(corr18mtx,"PM Peak")
thepmmc = CreateMatrixCurrency(corr18mtx,"PM Peak",null,null,)
EvaluateMatrixExpression(thepmmc,aexp,null,null,)
addmatrixcore(corr18mtx,"OFF Peak")
theopmc = CreateMatrixCurrency(corr18mtx,"OFF Peak",null,null,)
EvaluateMatrixExpression(theopmc,aexp,null,null,)

MergeMatrixElements(theammc,{ammc},null,null,{{"Force Missing","No"}})
MergeMatrixElements(thepmmc,{pmmc},null,null,{{"Force Missing","No"}})
MergeMatrixElements(theopmc,{opmc},null,null,{{"Force Missing","No"}})

amexp = "[Corr18 AM Peak] + [AM Peak]"
addmatrixcore(corr18mtx,"Final AM Peak")
thefammc = CreateMatrixCurrency(corr18mtx,"Final AM Peak",null,null,)
EvaluateMatrixExpression(thefammc,amexp,null,null,)

pmexp = "[Corr18 PM Peak] + [PM Peak]"
addmatrixcore(corr18mtx,"Final PM Peak")
thefpmmc = CreateMatrixCurrency(corr18mtx,"Final PM Peak",null,null,)
EvaluateMatrixExpression(thefpmmc,pmexp,null,null,)

opexp = "[Corr18 OFF Peak] + [OFF Peak]"
addmatrixcore(corr18mtx,"Final OFF Peak")
thefopmc = CreateMatrixCurrency(corr18mtx,"Final OFF Peak",null,null,)
EvaluateMatrixExpression(thefopmc,opexp,null,null,)

// trucks

atmtx = openmatrix(thepath+"\\hwy\\ODME_OD13.MTX","True")
tmc = CreateMatrixCurrency(atmtx,"ODME",null,null,)

trkmtx = copymatrix(tmc,{{"File Name",thepath+"\\hwy\\todtruck.mtx"},
{"Label","Truck TOD"},
{"File Based","Yes"}})

addmatrixcore(trkmtx,"AM Peak")
addmatrixcore(trkmtx,"PM Peak")
addmatrixcore(trkmtx,"OFF Peak")
ammc = CreateMatrixCurrency(trkmtx,"AM Peak",null,null,)
amexp = "[ODME] * 0.1714"
EvaluateMatrixExpression(ammc,amexp,null,null,)

pmmc = CreateMatrixCurrency(trkmtx,"PM Peak",null,null,)
pmexp = "[ODME] * 0.1655"
EvaluateMatrixExpression(pmmc,pmexp,null,null,)

```



```

    opmc = CreateMatrixCurrency(trkmtx,"OFF Peak",null,null,)
    opexp = "[ODME] * 0.6633"
    EvaluateMatrixExpression(opmc,opexp,null,null,)
    runmacro("closematrices",)
endMacro

macro "truckassign" (in_value)
    thepath = in_value[1]
    linefilename = in_value[2]
    linevw = in_value[3]
    thenetwork = in_value[4]
    trkcore = in_value[5]
    if trkcore = "OFF Peak" then capfield = "AB_CAP" else capfield = "AB_CAPPEAK"

    afilename = thepath+"\\hwy\\truck"+trkcore+"_asn_link.bin"
    if trkcore = "AM Peak" then thefields = {"AM_TRK_VOL"}
    if trkcore = "PM Peak" then thefields = {"PM_TRK_VOL"}
    if trkcore = "OFF Peak" then thefields = {"OP_TRK_VOL"}
    RunMacro("addfields",{thefields,linevw})
    RunMacro("TCB Init")
// STEP 1: Assignment
    Opts = {"Input",      {"Database",      linefilename},
           {"Network",   {"Network",      thenetwork},
           {"OD Matrix Currency", {"OD Matrix Currency", thepath+"\\hwy\\todtruck.MTX",
                               trkcore,
                               "NodeID",
                               "NodeID"}}}},
           {"Field",     {"FF Time",      "FFTIME"},
           {"Capacity",  {"Capacity",   capfield},
           {"Alpha",     {"Alpha",      "None"},
           {"Beta",      {"Beta",      "None"}}}},
           {"Global",    {"Load Method",  5},
           {"Alpha Value", {"Alpha Value", 0.15},
           {"Beta Value", {"Beta Value", 4},
           {"Iterations", {"Iterations", 20},
           {"Convergence", {"Convergence", 0.01}}}},
           {"Flag",      {"Do Emission", 0},
           {"Do Tabulation", {"Do Tabulation", 0}}}},
           {"Output",    {"Flow Table",  afilename}}}}

    if !RunMacro("TCB Run Procedure", "Assignment", Opts) then Return(RunMacro("TCB Closing", 0))

    asgvw = Opentable("truck_asgn","FFB",{afilename,})
    jnvw = JoinViews(linevw+asgvw,linevw+".ID",asgvw+".ID1",)

    arec = GetFirstRecord(jnvw+"|",null)
    while (arec <> null) do
        jnvw.(thefields[1]) = jnvw.Tot_Flow
        arec = GetNextRecord(jnvw+"|",null,null)
    end
    closeview(asgvw)
    return(jnvw)
endMacro

Macro "assign" (in_value)
    thepath = in_value[1]

```

```

linefilename = in_value[2]
  linevw = in_value[3]
thenetwork = in_value[4]
todcore = in_value[5]
afilename = thepath+"\\hwy\\"+todcore+"_asn_link.bin"
if todcore = "Final OFF Peak" then capfield = "AB_CAP" else capfield = "AB_CAPPEAK"

if todcore = "Final AM Peak" then thefields = {"AM_AUTO_VOL"}
  if todcore = "Final PM Peak" then thefields = {"PM_AUTO_VOL"}
  if todcore = "Final OFF Peak" then thefields = {"OP_AUTO_VOL"}
  RunMacro("addfields",{thefields,linevw})
RunMacro("TCB Init")
// STEP 1: Assignment
  Opts = {{"Input",      {"Database",      linefilename},
          {"Network",   thenetwork},
          {"OD Matrix Currency", {thepath+"\\tod\\todcorr18.mtx",
                                todcore,
                                "From",
                                "To"}}}},
          {"Field",     {"FF Time",      "FFTIME"},
          {"Capacity",  capfield},
          {"Alpha",     "None"},
          {"Beta",      "None"},
          {"Preload",   "AB_FLOW"}}},
          {"Global",    {"Load Method",   5},
          {"Alpha Value", 0.15},
          {"Beta Value",  4},
          {"Iterations",  30},
          {"Convergence", 0.01}}},
          {"Flag",      {"Do Emission",  0},
          {"Do Tabulation", 0}}},
          {"Output",    {"Flow Table",   afilename}}}}

  if !RunMacro("TCB Run Procedure", "Assignment", Opts) then Return( RunMacro("TCB Closing",
0) )

  asgvw = Opentable("auto_asgn","FFB",{afilename,})
  jnvw = JoinViews(linevw+asgvw,linevw+".ID",asgvw+".ID1",)

  arec = GetFirstRecord(jnvw+"|",null)
  while (arec <> null) do
    jnvw.(thefields[1]) = jnvw.Tot_Flow
    arec = GetNextRecord(jnvw+"|",null,null)
  end
  closeview(asgvw)
  closeview(jnvw)

endMacro

macro "finalcalcs" (in_value)
  linevw = in_value[1]
  thefields = {"Dly_Auto_Vol","Dly_Trk_Vol","Dly_Tot_Vol"}
  runmacro("addfields",{thefields,linevw})
  arec = GetFirstRecord(linevw+"|",null)
  while (arec <> null) do
    linevw.Dly_Auto_Vol = linevw.AM_Auto_Vol + linevw.PM_Auto_Vol + linevw.OP_Auto_Vol
    linevw.Dly_Trk_Vol = linevw.AM_Trk_vol + linevw.PM_Trk_vol + linevw.OP_Trk_Vol
    linevw.Dly_Tot_Vol = linevw.Dly_Auto_Vol + linevw.Dly_Trk_Vol
  end

```



```
        arec = GetNextRecord(linevw+"|",null,null)
    end
endMacro

//*****Below are some utility macros *****

macro "closematrices"
    matrixlist = GetMatrices()
    if (matrixlist.length > 0) then do
        allmatrices = matrixlist[1]
        for m = 1 to allmatrices.length do
            closematrix(allmatrices[m])
        end
    end
endMacro

macro "create_table" (invalue)
// This macro creates a bunch of tables based on the need.  Three values
// are passed to this macro, a view name, a file name ( no path ), and a type of table

    atest = invalue[3]
    if atest = "trip" then do
        CreateTable(invalue[1],invalue[4]+"\\"+invalue[2],"DBASE",{
            {"TAZID","Integer",5,0,"Yes"},
            {"SZ","Integer",5,0,"No"},
            {"HHS1AO0","Real",10,1,"No"},
            {"HHS1AO1","Real",10,1,"No"},
            {"HHS1AO2","Real",10,1,"No"},
            {"HHS1AO3","Real",10,1,"No"},
            {"HHS2AO0","Real",10,1,"No"},
            {"HHS2AO1","Real",10,1,"No"},
            {"HHS2AO2","Real",10,1,"No"},
            {"HHS2AO3","Real",10,1,"No"},
            {"HHS3AO0","Real",10,1,"No"},
            {"HHS3AO1","Real",10,1,"No"},
            {"HHS3AO2","Real",10,1,"No"},
            {"HHS3AO3","Real",10,1,"No"},
            {"HHS4AO0","Real",10,1,"No"},
            {"HHS4AO1","Real",10,1,"No"},
            {"HHS4AO2","Real",10,1,"No"},
            {"HHS4AO3","Real",10,1,"No"},
            {"Total","Real",10,0,"No"},
            {"Int_Total","Real",10,0,"No"}}})
    end
    if atest = "hhdist" then do
        CreateTable(invalue[1],invalue[4]+"\\"+invalue[2],"DBASE",{
            {"TAZID","Integer",5,0,"Yes"},
            {"SZ","Integer",5,0,"No"},
            {"HHS1AO0","Real",10,1,"No"},
            {"HHS1AO1","Real",10,1,"No"},
            {"HHS1AO2","Real",10,1,"No"},
            {"HHS1AO3","Real",10,1,"No"},
            {"HHS2AO0","Real",10,1,"No"},
            {"HHS2AO1","Real",10,1,"No"},
            {"HHS2AO2","Real",10,1,"No"},
            {"HHS2AO3","Real",10,1,"No"},
            {"HHS3AO0","Real",10,1,"No"},
            {"HHS3AO1","Real",10,1,"No"},
            {"HHS3AO2","Real",10,1,"No"},
            {"HHS3AO3","Real",10,1,"No"},
            {"HHS4AO0","Real",10,1,"No"},
            {"HHS4AO1","Real",10,1,"No"},
            {"HHS4AO2","Real",10,1,"No"},
            {"HHS4AO3","Real",10,1,"No"},
            {"Total","Real",10,0,"No"},
            {"Int_Total","Real",10,0,"No"}}})
    end
endMacro
```

```

        {"HHS2AO3", "Real", 10, 1, "No"},
        {"HHS3AO0", "Real", 10, 1, "No"},
        {"HHS3AO1", "Real", 10, 1, "No"},
        {"HHS3AO2", "Real", 10, 1, "No"},
        {"HHS3AO3", "Real", 10, 1, "No"},
        {"HHS4AO0", "Real", 10, 1, "No"},
        {"HHS4AO1", "Real", 10, 1, "No"},
        {"HHS4AO2", "Real", 10, 1, "No"},
        {"HHS4AO3", "Real", 10, 1, "No"}))

```

end

if atest = "prod" then do

```

    CreateTable(invalue[1], invalue[4] + "\\ " + invalue[2], "DBASE", {
        {"TAZID", "Integer", 5, 0, "Yes"},
        {"HBW", "Integer", 10, 0, "No"},
        {"NHB", "Integer", 10, 0, "No"},
        {"HBO", "Integer", 10, 0, "No"},
        {"LNG", "Integer", 10, 0, "No"}})

```

end

if atest = "attract" then do

```

    CreateTable(invalue[1], invalue[4] + "\\ " + invalue[2], "DBASE", {
        {"TAZID", "Integer", 5, 0, "Yes"},
        {"HBW", "Integer", 10, 0, "No"},
        {"NHB", "Integer", 10, 0, "No"},
        {"HBO", "Integer", 10, 0, "No"},
        {"LNG", "Integer", 10, 0, "No"}})

```

end

hh = OpenTable(invalue[1], "DBASE", {invalue[4] + "\\ " + invalue[2],})

return (hh)

endMacro

macro "addfields" (in\_value)

fldnames = in\_value[1]

struct = GetTableStructure(in\_value[2])

viewflds = getFields(in\_value[2], numeric)

for i=1 to struct.length do

```

    struct[i]=struct[i]+{struct[i][1]}

```

end

for i=1 to fldnames.length do

```

    pos = ArrayPosition(viewflds[1], {fldnames[i],})

```

if pos = 0 then do

```

    newstr = newstr + {{fldnames[i], "Real", 10, 3, "false", null, null, null, null}}

```

modtab = 1

end

end

if modtab = 1 then do

```

    newstr = struct+newstr

```

```

    ModifyTable(in_value[2], newstr)

```

end

endMacro



```
// *****Create Transit Routes, network and Skims. Not Currently Used --
Caliper Needs to Debug *****8
// rtes = RunMacro("createroutes",{thepath,cnet,linefilename,linevw})
// RunMacro("buffers",{linevw,nodevw})
// trannet = RunMacro("transitnetwork",{thepath,linevw})
// runmacro("transkim")
Macro "createroutes"
  RunMacro("TCB Init")
// STEP 1: Create RS From Table
  Opts = {{ "Input",      {{ "Network",
"D:\\PROJECTS\\6956\\1998BASE\\HWY\\HIGHWAY_INTNL.NET"},
          {{ "Link Set",
"D:\\Projects\\6956\\1998Base\\hwy\\I69_base_2000.DBD|Base_2000 Links",
          "Base_2000 Links"}}},
          {{ "Tour Table",      {"D:\\PROJECTS\\6956\\STOPTABLE.DBF"}}}},
  {"Global",  {{ "Cost Field",      1},
               {{ "Route ID Field", 3},
               {{ "Node ID Field",  4},
               {{ "Include Stop",   1},
               {{ "Stop Flag Field", 2},
               {{ "User ID Field",  4}}}},
  {"Output",  {{ "Output Routes",
"D:\\Projects\\6956\\1998Base\\routes\\test.rts"}}}}
  if !RunMacro("TCB Run Operation", "Create RS From Table", Opts) then Return( RunMacro("TCB
Closing", 0) )
  else
  RunMacro("TCB Closing", 1)
endMacro

Macro "buffers"
  setlayer("Base_2000 Links")
  linkselect =SelectByVicinity("17mirad", "Several", "Stops|", 15.0)
  //showmessage(string(linkselect))
  setlayer("Base_2000 Node")
  tazselect = SelectByVicinity("Selection", "Several", "Stops|", 13.0)
  aQuery = "Select * where taz < 743"
  SelectByQuery("Selection","Subset",aQuery,)
endMacro

Macro "transitnetwork"
rfllds = {{ "Route_id","Vehicle Routes.Route_id"}}

lfllds = {{ "Length","Base_2000 Links.Length"},
          {{ "FFtime","Base_2000 Links.FFTime"},
          {{ "Ctime","Base_2000 Links.CTime"}}

dim llflds[lfllds.length]
  for j = 1 to lfllds.length do
    llflds[j] = lfllds[j]+{"SUMFRAC"}
  end

opts = {{ "Route Attributes",  rfllds},
        {{ "Street Attributes", lfllds},
        {{ "Walk","Yes"},
        {{ "Snap Dist",0.05},
        {{ "Link Direction Field","Base_2000 Links.Dir"},
```

```
    {"Link Attributes",    llflds},
    {"Snap Stops", "Yes"},
    {"Snap Stop Distance", 0.05}}
```

```
showarray(rflds)
showarray(lflds)
showarray(opts)
curversion = RunMacro("TC30 Get Transit Network Version")
setlayer("Vehicle Routes")
thenetwork = createTransitNetwork(,"Stops","Base_2000
Links|17mirad","d:\\projects\\6956\\1998base\\routes\\98tran.tnw",,opts)
SetNetworkInformationItem(thenetwork,"Version",{curversion})
runmacro("transkim")
endMacro
```

```
Macro "transkim"
    RunMacro("TCB Init")
// STEP 1: Transit Skim
    Opts = {{"Input",      {"Database",
"D:\\Projects\\6956\\1998Base\\hwy\\I69_base_2000.DBD"},
           {"Network",
"D:\\Projects\\6956\\1998Base\\routes\\98tran.tnw"},
           {"Origin Set",
{"D:\\Projects\\6956\\1998Base\\hwy\\I69_base_2000.DBD|Base_2000 Node",
"Base_2000 Node",
"Selection"}},
           {"Destination Set",
{"D:\\Projects\\6956\\1998Base\\hwy\\I69_base_2000.DBD|Base_2000 Node",
"Base_2000 Node",
"Selection"}},},
    {"Global",      {"SP Method",      1},
                    {"Skim Var",      {6,
5,
7,
8}},
                    {"OD Layer Type", 2}},
    {"Output",      {"Skim Matrix",    {"Label",
"Skim Matrix (Shortest Path)",
"File Name",
"D:\\Projects\\6956\\1998Base\\routes\\TR_SKIM5.MTX" }}}}}

    if !RunMacro("TCB Run Procedure", "Transit Skim", Opts) then goto quit
done:
Return( RunMacro("TCB Closing", 1) )
quit:
Return( RunMacro("TCB Closing", 0) )
endMacro
```



# Appendix C

## *2025 Updated Indiana Model GIS DK Program*



```
Dbox "startdbx",,50 title: "4 Step Model Process"  ToolBox
  init do
    runmacro("closematrices",)
    layer_names = GetLayerNames()
    theviews = GetViewNames()
    todlist = {"AM Peak","Final AM Peak"}, {"PM Peak","Final PM Peak"}, {"OFF Peak","Final OFF Peak"}}
  enditem
// Create the Left Side of the Dialog Box First

Frame 1,0.5,23,19 Prompt: "General Settings"
Text "Line Layer" 2.5, 1.75
Popdown Menu "Lines" 2.5, 3
  List: layer_names
  variable: linevwidx
  do
    linevw = layer_names[linevwidx]
    lineinfo = GetLayerInfo(linevw)
    linefilename = lineinfo[10]
  enditem

Text "Node Layer" 2.5, 4.75
Popdown Menu "Nodes" 2.5, 6
  List: layer_names
  variable: nodevwidx
  do
    nodevw = layer_names[nodevwidx]
    nodeinfo = GetLayerInfo(nodevw)
    nodefilename = nodeinfo[10]
  enditem

text "TAZ Layer" 2.5,7.75
Popdown Menu "Taz" 2.5, 9
  List: layer_names
  variable: tazvwidx
  do
    tazvw = layer_names[tazvwidx]
    tazinfo = GetLayerInfo(tazvw)
    tazfilename = tazinfo[10]
  enditem
```



```
text "Indiana Zones" 2.5,10.75
Edit Int "internals" 2.5, 12
    variable: intzones
```

```
text "External Area" 2.5,13.75
Edit Int "internals" 2.5, 15
    variable: extzones
```

```
text "Total Zones" 2.5,16.75
Edit Int "allzones" 2.5, 18
    variable: allzones
```

```
// Now create the right side of the Dialog Box
```

```
Frame 25, 0.5 , 24, 16 Prompt: "Modeling Modules"
```

```
button "Trip Generation" 27,2,20 do
```

```
    hhdist = runMacro("create_hhdist", {theviews, thepath, tazvw, extzones})
```

```
enditem
```

```
button "Trip Distribution" 27,4.5,20 do
```

```
    RunMacro("balance_ps_to_as", {thepath, tazvw, tazfilename})
```

```
    zonelist = {intzones, extzones}
```

```
    for a = 1 to 2 do
```

```
        zones = zonelist[a]
```

```
        cnet = runMacro("create_hnet", {thepath, linevw, nodevw, zones, a})
```

```
        impmat = RunMacro("Multi_Path", {thepath, nodevw, tazvw, linefilename, zones, a, cnet})
```

```
        RunMacro("friction_factors", {thepath, impmat, a})
```

```
        RunMacro("TripDist", {thepath, tazvw, tazfilename, a, intzones})
```

```
    end
```

```
enditem
```

```
button "Mode Choice" 27,7,20 do
```

```
    RunMacro("thematrix", thepath)
```

```
    RunMacro("mnl_eval_lng", {thepath, tazvw, tazfilename, intzones})
```

```
    runMacro("test_matrix", thepath)
```

```
enditem
```



```
button "Time of Day" 27, 9.5,20 do
  runMacro("matrix_manip",thepath)
enditem

button "Traffic Assignment" 27, 12, 20 do
  for t = 1 to todlist.length do
    trucknet = RunMacro("create_hnet",{thepath,linevw,nodevw,allzones,2})
    ajvw = runmacro("truckassign",{thepath,linefilename,linevw,trucknet,todlist[t][1]})
    finalnet = RunMacro("create_hnet",{thepath,ajvw,nodevw,allzones,3})
    closeview(ajvw)
    runmacro("assign",{thepath,linefilename,linevw,finalnet,todlist[t][2]})
  end
  runmacro("finalcalcs",{linevw})
enditem

button "Run All Steps" 27,14.5,20 do
  hhdist = runMacro("create_hhdist",{theviews,thepath,tazvw,extzones})
  RunMacro("balance_ps_to_as",{thepath,tazvw,tazfilename})
  zonelist = {intzones,extzones}
  for a = 1 to 2 do
    zones = zonelist[a]
    cnet = runMacro("create_hnet",{thepath,linevw,nodevw,zones,a})
    impmat = RunMacro("Multi_Path",{thepath,nodevw,tazvw,linefilename,zones,a,cnet})
    RunMacro("friction_factors",{thepath,impmat,a})
    RunMacro("TripDist",{thepath,tazvw,tazfilename,a,intzones})
  end
  RunMacro("thematrix",thepath)
  RunMacro("mnl_eval_lng",{thepath,tazvw,tazfilename,intzones})
  RunMacro("test_matrix",thepath)
  runMacro("matrix_manip",thepath)
  for t = 1 to todlist.length do
    trucknet = RunMacro("create_hnet",{thepath,linevw,nodevw,allzones,2})
    ajvw = runmacro("truckassign",{thepath,linefilename,linevw,trucknet,todlist[t][1]})
    finalnet = RunMacro("create_hnet",{thepath,ajvw,nodevw,allzones,3})
    closeview(ajvw)
    runmacro("assign",{thepath,linefilename,linevw,finalnet,todlist[t][2]})
  end
  runmacro("finalcalcs",{linevw})
enditem
```



```
// Now create the Path part of the Dialog box on the bottom

Frame 1, 20 , 48, 3 Prompt: "Modeling Path"
button "Browse" 2.5,21.5, 10 do
    thepath = ChooseDirectory("Choose the Modeling Directory")
enditem

text "Model Path" 15.5,21.5,32 Framed variable: thepath

Frame 1,23,48,2.5
text "Alternative Code" 10.5,24,15
Edit Text "Alternative" 26,24,10
    variable: alt

// Now put the cancel button on the bottom of the dialog box so we can get out of it

button "Cancel" 17.5, 27, 15 do return() enditem
endDbox

Dbox "Enter_Value" (in_value) Title: in_value[2]

    //Text 1,1
    Edit Text "num iter item" 1,1,30
    //prompt: "Please enter path of Model Files"
    Variable: thepath
    do
        Return(thepath)
    enditem
endDbox

Dbox "selectview" (in_value) Title: in_value[1]
    init do
        if in_value[2].length = 0 then do
            ShowMessage("There are no datafiles loaded")
        return()
        field_idx = 1
        end
    enditem

    scroll list 2.5,1,25,10
```



```
list: in_value[2]
Variables: field_idx

    button "OK" 30, 1, 9 do
Return(field_idx)
enditem
button "Cancel"30, 3, 9 do
    return ()
enditem
endDbox

//***** This Macro Creates the Highway Network *****

Macro "create_hnet" (in_value)
thepath = in_value[1]
vw = in_value[2]
nodes = in_value[3]
zones = string(in_value[4])
flag = in_value[5]
setview(vw)
if flag = 1 then do
    nettype = "int"
    thenetwork = CreateNetwork(null,thepath+"\\hwy\\highway_"+nettype+".net","Highway Network",
    {{"FFTIME","FFTIME"},{"Length","Length"},{"CTIME","CTIME"},{"AB_CAPPEAK","AB_CAPPEAK"},{"BA_CAPPEAK","BA_CAPPEAK"},
},
    {"AB_CAP","AB_CAP"},{"BA_CAP","BA_CAP"},,,)
    end
if flag = 2 then do
    nettype = "intext"
    thenetwork = CreateNetwork(null,thepath+"\\hwy\\highway_"+nettype+".net","Highway Network",
    {{"FFTIME","FFTIME"},{"Length","Length"},{"CTIME","CTIME"},{"AB_CAPPEAK","AB_CAPPEAK"},{"BA_CAPPEAK","BA_CAPPEAK"},
},
    {"AB_CAP","AB_CAP"},{"BA_CAP","BA_CAP"},,,)
    end
if flag = 3 then do
    nettype = "final"
    thenetwork = CreateNetwork(null,thepath+"\\hwy\\highway_"+nettype+".net","Highway Network",
```



```
},
    {"FFTIME", "FFTIME"}, {"Length", "Length"}, {"CTIME", "CTIME"}, {"AB_CAPPEAK", "AB_CAPPEAK"}, {"BA_CAPPEAK", "BA_CAPPEAK"},
    {"AB_CAP", "AB_CAP"}, {"BA_CAP", "BA_CAP"}, {"AB_FLOW", "AB_FLOW"}, {"BA_FLOW", "BA_FLOW"}}, , ,)

    end
    nodelayer = nodes
    SetLayer(nodelayer)
    nsetname = "taz"
    qry = "Select * where taz <="+zones
    n = SelectByQuery(nsetname, "Several", qry,)
    SetCursor("Hourglass")
    idxSet = nodelayer + "|" + nsetname
    opts = {"Use Centroids", "True"},
           {"Use Turn Penalties", "False"},
           {"Centroids Set", idxSet}}

    ChangeNetworkSettings(thenetwork, opts)
    netinfo = GetNetworkInfo(thenetwork)
    netfile = netinfo[1]
    return(netfile)
endMacro

//***** END Create Highway Macro *****

// *****This Is the Trip Generation Section *****

// *****This macro creates the Household Distribution *****

macro "create_hhdist" (in_value)
    theviews = in_value[1]
    thepath = in_value[2]
    tzvw = in_value[3]
    intzone = in_value[4]

    distvw = runmacro("create_table", {"temp_hh_dist", "temp_hh.dbf", "hhdist", thepath+"\\tg"})
    pctvw = opentable("percents", "DATABASE", {thepath+"\\tg\\pcts_exp.dbf", })
    stwpctvw = opentable("state_pcts", "DATABASE", {thepath+"\\tg\\swtpct.dbf", })
```



```
pcttazvw = JoinViews("TAZ+PCTS",tzvw+".TAZ_ID",pctvw+".TAZ_ID1",)  
//thequery = "Select * where ID <= "+intzones  
//setview(pcttazvw)  
//SelectByQuery("TAZS","Several",thequery,)  
tzrec = GetFirstRecord(pcttazvw+"|",null)  
// declare variables
```

```
totalhh = 0  
ttlhh1ao0 = 0  
ttlhh1ao1 = 0  
ttlhh1ao2 = 0  
ttlhh1ao3 = 0  
ttlhh2ao0 = 0  
ttlhh2ao1 = 0  
ttlhh2ao2 = 0  
ttlhh2ao3 = 0  
ttlhh3ao0 = 0  
ttlhh3ao1 = 0  
ttlhh3ao2 = 0  
ttlhh3ao3 = 0  
ttlhh4ao0 = 0  
ttlhh4ao1 = 0  
ttlhh4ao2 = 0  
ttlhh4ao3 = 0
```

```
while tzrec <> null do  
  tazid = pcttazvw.("TAZ_ID")  
  sz = pcttazvw.("SZ")  
  if sz <> null then do  
    hholds = pcttazvw.("HOUSEHOLDS")  
    totalhh = hholds+totalhh  
    pcthh1 = pcttazvw.("PCTHHS1")  
    pcthh2 = pcttazvw.("PCTHHS2")  
    pcthh3 = pcttazvw.("PCTHHS3")  
    pcthh4 = pcttazvw.("PCTHHS4")  
    pctA00 = pcttazvw.("PCTA00")  
    pctA01 = pcttazvw.("PCTA01")  
    pctA02 = pcttazvw.("PCTA02")  
    pctA03 = pcttazvw.("PCTA03")  
    hh1ao0 = hholds*pcthh1*pctA00  
    hh1ao1 = hholds*pcthh1*pctA01
```



```
hh1ao2 = hholds*pcthh1*pctA02
hh1ao3 = hholds*pcthh1*pctA03
hh2ao0 = hholds*pcthh2*pctA00
hh2ao1 = hholds*pcthh2*pctA01
hh2ao2 = hholds*pcthh2*pctA02
hh2ao3 = hholds*pcthh2*pctA03
hh3ao0 = hholds*pcthh3*pctA00
hh3ao1 = hholds*pcthh3*pctA01
hh3ao2 = hholds*pcthh3*pctA02
hh3ao3 = hholds*pcthh3*pctA03
hh4ao0 = hholds*pcthh4*pctA00
hh4ao1 = hholds*pcthh4*pctA01
hh4ao2 = hholds*pcthh4*pctA02
hh4ao3 = hholds*pcthh4*pctA03
//if hh1ao0 = null then do
```

```
//showarray( {tazid,sz,totalhh,pcthh1,pcthh2,pcthh3,pcthh4,pctA00,pctA01,pctA02,pctA03,hh1ao0,hh1ao1,hh1ao2,
              //hh1ao3,hh2ao0,hh2ao1,hh2ao2,hh2ao3,hh3ao0,hh3ao1,hh3ao2,hh3ao3,hh4ao0,hh4ao1,hh4ao2,hh4ao3} )
//end
ttlhh1ao0 = hh1ao0 + ttlhh1ao0
ttlhh1ao1 = hh1ao1 + ttlhh1ao1
ttlhh1ao2 = hh1ao2 + ttlhh1ao2
ttlhh1ao3 = hh1ao3 + ttlhh1ao3
ttlhh2ao0 = hh2ao0 + ttlhh2ao0
ttlhh2ao1 = hh2ao1 + ttlhh2ao1
ttlhh2ao2 = hh2ao2 + ttlhh2ao2
ttlhh2ao3 = hh2ao3 + ttlhh2ao3
ttlhh3ao0 = hh3ao0 + ttlhh3ao0
ttlhh3ao1 = hh3ao1 + ttlhh3ao1
ttlhh3ao2 = hh3ao2 + ttlhh3ao2
ttlhh3ao3 = hh3ao3 + ttlhh3ao3
ttlhh4ao0 = hh4ao0 + ttlhh4ao0
ttlhh4ao1 = hh4ao1 + ttlhh4ao1
ttlhh4ao2 = hh4ao2 + ttlhh4ao2
ttlhh4ao3 = hh4ao3 + ttlhh4ao3
addrecords(distvw,
{ "TAZID", "SZ", "HHS1A00", "HHS1A01", "HHS1A02", "HHS1A03", "HHS2A00", "HHS2A01", "HHS2A02",
  "HHS2A03", "HHS3A00", "HHS3A01", "HHS3A02", "HHS3A03", "HHS4A00", "HHS4A01",
  "HHS4A02", "HHS4A03" }, { {tazid,sz,hh1ao0,hh1ao1,hh1ao2,hh1ao3,hh2ao0,hh2ao1,hh2ao2,
  hh2ao3,hh3ao0,hh3ao1,hh3ao2,hh3ao3,hh4ao0,hh4ao1,hh4ao2,hh4ao3} }, null )
```

end



```
tzrec = GetNextRecord(pcttavzw+"|",null,null)
end

//showarray({ttlhh1ao0,ttlhh1ao1,ttlhh1ao2,ttlhh1ao3,ttlhh2ao0,ttlhh2ao1,ttlhh2ao2,ttlhh2ao2,ttlhh3ao0,ttlhh3ao1,
//          ttlhh3ao2,ttlhh3ao3,ttlhh4ao0,ttlhh4ao1,ttlhh4ao2,ttlhh4ao3})

closeview(pcttavzw)
finaldistvw = runmacro("create_table",{ "final_hh_dist", "final_hh.dbf", "hhdist", thepath+"\\tg" })

// do adjustment to statewide percentages here

swprec = GetFirstRecord(stwpctvw+"|",null)
while swprec <> null do
    swhh1ao0 = stwpctvw.("HHS1AO0")
    swhh1ao1 = stwpctvw.("HHS1AO1")
    swhh1ao2 = stwpctvw.("HHS1AO2")
    swhh1ao3 = stwpctvw.("HHS1AO3")
    swhh2ao0 = stwpctvw.("HHS2AO0")
    swhh2ao1 = stwpctvw.("HHS2AO1")
    swhh2ao2 = stwpctvw.("HHS2AO2")
    swhh2ao3 = stwpctvw.("HHS2AO3")
    swhh3ao0 = stwpctvw.("HHS3AO0")
    swhh3ao1 = stwpctvw.("HHS3AO1")
    swhh3ao2 = stwpctvw.("HHS3AO2")
    swhh3ao3 = stwpctvw.("HHS3AO3")
    swhh4ao0 = stwpctvw.("HHS4AO0")
    swhh4ao1 = stwpctvw.("HHS4AO1")
    swhh4ao2 = stwpctvw.("HHS4AO2")
    swhh4ao3 = stwpctvw.("HHS4AO3")
    swprec = GetNextRecord(stwpctvw+"|",null,null)
end

adjfhh1ao0 = (swhh1ao0*totalhh) / ttlhh1ao0
adjfhh1ao1 = (swhh1ao1*totalhh) / ttlhh1ao1
adjfhh1ao2 = (swhh1ao2*totalhh) / ttlhh1ao2
adjfhh1ao3 = (swhh1ao3*totalhh) / ttlhh1ao3
adjfhh2ao0 = (swhh2ao0*totalhh) / ttlhh2ao0
adjfhh2ao1 = (swhh2ao1*totalhh) / ttlhh2ao1
adjfhh2ao2 = (swhh2ao2*totalhh) / ttlhh2ao2
adjfhh2ao3 = (swhh2ao3*totalhh) / ttlhh2ao3
adjfhh3ao0 = (swhh3ao0*totalhh) / ttlhh3ao0
adjfhh3ao1 = (swhh3ao1*totalhh) / ttlhh3ao1
```



```
adjfhh3ao2 = (swhh3ao2*totalhh) / ttlhh3ao2  
adjfhh3ao3 = (swhh3ao3*totalhh) / ttlhh3ao3  
adjfhh4ao0 = (swhh4ao0*totalhh) / ttlhh4ao0  
adjfhh4ao1 = (swhh4ao1*totalhh) / ttlhh4ao1  
adjfhh4ao2 = (swhh4ao2*totalhh) / ttlhh4ao2  
adjfhh4ao3 = (swhh4ao3*totalhh) / ttlhh4ao3
```

```
tmpdistrec = GetFirstRecord(distvw+"|",null)
```

```
while tmpdistrec <> null do  
  tazid2 = distvw.("TAZID")  
  sz2 = distvw.("SZ")  
  temphh1a0 = distvw.("HHS1A00")  
  adjhh1ao0 = temphh1a0 * adjfhh1ao0  
  temphh1a1 = distvw.("HHS1A01")  
  adjhh1ao1 = temphh1a1 * adjfhh1ao1  
  temphh1a2 = distvw.("HHS1A02")  
  adjhh1ao2 = temphh1a2 * adjfhh1ao2  
  temphh1a3 = distvw.("HHS1A03")  
  adjhh1ao3 = temphh1a3 * adjfhh1ao3  
  temphh2a0 = distvw.("HHS2A00")  
  adjhh2ao0 = temphh2a0 * adjfhh2ao0  
  temphh2a1 = distvw.("HHS2A01")  
  adjhh2ao1 = temphh2a1 * adjfhh2ao1  
  temphh2a2 = distvw.("HHS2A02")  
  adjhh2ao2 = temphh2a2 * adjfhh2ao2  
  temphh2a3 = distvw.("HHS2A03")  
  adjhh2ao3 = temphh2a3 * adjfhh2ao3  
  temphh3a0 = distvw.("HHS3A00")  
  adjhh3ao0 = temphh3a0 * adjfhh3ao0  
  temphh3a1 = distvw.("HHS3A01")  
  adjhh3ao1 = temphh3a1 * adjfhh3ao1  
  temphh3a2 = distvw.("HHS3A02")  
  adjhh3ao2 = temphh3a2 * adjfhh3ao2  
  temphh3a3 = distvw.("HHS3A03")  
  adjhh3ao3 = temphh3a3 * adjfhh3ao3  
  temphh4a0 = distvw.("HHS4A00")  
  adjhh4ao0 = temphh4a0 * adjfhh4ao0  
  temphh4a1 = distvw.("HHS4A01")  
  adjhh4ao1 = temphh4a1 * adjfhh4ao1  
  temphh4a2 = distvw.("HHS4A02")
```



```
adjhh4ao2 = temp hh4a2 * adjfhh4ao2
temp hh4a3 = distvw.( "HHS4AO3" )
adjhh4ao3 = temp hh4a3 * adjfhh4ao3
addrecords( finaldistvw,
{ "TAZID", "SZ", "HHS1AO0", "HHS1AO1", "HHS1AO2", "HHS1AO3", "HHS2AO0", "HHS2AO1", "HHS2AO2",
  "HHS2AO3", "HHS3AO0", "HHS3AO1", "HHS3AO2", "HHS3AO3", "HHS4AO0", "HHS4AO1",
  "HHS4AO2", "HHS4AO3" }, { { tazid2, sz2, adjhh1ao0, adjhh1ao1, adjhh1ao2, adjhh1ao3, adjhh2ao0, adjhh2ao1, adjhh2ao2,
  adjhh2ao3, adjhh3ao0, adjhh3ao1, adjhh3ao2, adjhh3ao3, adjhh4ao0, adjhh4ao1, adjhh4ao2, adjhh4ao3 } }, null )
tmpdistrec =  GetNextRecord( distvw+"|", null, null )
end
closeview( distvw )
//createeditor( "final_hh_dist", finaldistvw+"|", null, )
deletefile( thepath+"\\tg\\temp_hh.dbf" )
deletefile( thepath+"\\tg\\temp_hh.mdX" )
prdvw = runmacro( "create_tripdist", { finaldistvw, theviews, thepath } )
attrmodel = runmacro( "attraction_model", { tzvw, prdvw, thepath } )
endmacro
// *****End Household distribution Macro *****

// *****This section creates the production rates *****
macro "create_tripdist" ( in_value )

  theviews = in_value[2]
  newdistvw = in_value[1]
  thepath = in_value[3]

  hbwprd = opentable( "hbw_prate", "DBASE", { thepath+"\\tg\\hbwprate_30.dbf", } )
  hbopr = opentable( "hbo_prate", "DBASE", { thepath+"\\tg\\hbopr_30.dbf", } )
  nhbprd = opentable( "nhb_prate", "DBASE", { thepath+"\\tg\\nhbprate_30.dbf", } )
  lngprd = opentable( "lng_prate", "DBASE", { thepath+"\\tg\\lngprate_30.dbf", } )
  trppct = opentable( "internal_pcts", "DBASE", { thepath+"\\tg\\intpcts_exp.dbf", } )

  prdlist = { hbwprd, hbopr, nhbprd, lngprd }
  triplist = { "hbw_trip", "hbo_trip", "nhb_trip", "lng_trip" }
  prodvw = runmacro( "create_table", { "productions", "product.dbf", "prod", thepath+"\\tg" } )
  for i = 1 to prdlist.length do

    prdvw = prdlist[i]
    triptab = triplist[i]
    purp = left( triptab, 3 )
    tripvw = runmacro( "create_table", { triptab, triptab+".dbf", "trip", thepath+"\\tg" } )
```



```
prec = GetFirstRecord(prdvw+"|",null)

while prec <> null do

    sz3 = prdvw.("SZ")
    aquery = "Select * where SZ = "+string(sz3)
    setview (trppct)
    acount = selectbyquery("trippct","Several",aquery,)

    if acount > 1 then do
        showmessage("Problem with trippct table!")
        return ()
    end

    pctrec = GetFirstRecord(trppct+"|trippct",null)

    while pctrec <> null do
        intpct = trppct.(purp+"pct")
        pctrec = GetNextRecord(trppct+"|trippct",null,null)
    end

    setview(newdistvw)

    SelectByQuery("Tony","Several",aquery,)
    trec = GetFirstRecord(newdistvw+"|Tony",null)

    while trec <> null do
        tazid3 = newdistvw.("TAZID")
        trphh1ao0 = prdvw.("HH1A00")* (newdistvw.("HHS1A00"))
        trphh1ao1 = prdvw.("HH1A01") * (newdistvw.("HHS1A01"))
        trphh1ao2 = prdvw.("HH1A02") * (newdistvw.("HHS1A02"))
        trphh1ao3 = prdvw.("HH1A03") * (newdistvw.("HHS1A03"))
        trphh2ao0 = prdvw.("HH2A00") * (newdistvw.("HHS2A00"))
        trphh2ao1 = prdvw.("HH2A01") * (newdistvw.("HHS2A01"))
        trphh2ao2 = prdvw.("HH2A02") * (newdistvw.("HHS2A02"))
        trphh2ao3 = prdvw.("HH2A03") * (newdistvw.("HHS2A03"))
        trphh3ao0 = prdvw.("HH3A00") * (newdistvw.("HHS3A00"))
        trphh3ao1 = prdvw.("HH3A01") * (newdistvw.("HHS3A01"))
        trphh3ao2 = prdvw.("HH3A02") * (newdistvw.("HHS3A02"))
        trphh3ao3 = prdvw.("HH3A03") * (newdistvw.("HHS3A03"))
```



```
trphh4ao0 = prdvw.("HH4AO0") * (newdistvw.("HHS4AO0"))
trphh4ao1 = prdvw.("HH4AO1") * (newdistvw.("HHS4AO1"))
trphh4ao2 = prdvw.("HH4AO2") * (newdistvw.("HHS4AO2"))
trphh4ao3 = prdvw.("HH4AO3") * (newdistvw.("HHS4AO3"))
totaltrps = trphh1ao0+trphh1ao1+trphh1ao2+trphh1ao3+trphh2ao0+trphh2ao1+trphh2ao2+trphh2ao3
           +trphh3ao0+trphh3ao1+trphh3ao2+trphh3ao3+trphh4ao0+trphh4ao1+trphh4ao2+trphh4ao3

totalintrips = totaltrps * intpct
TazQuery = "Select * where TAZID = "+string(tazid3)

setview(prodvw)
tzcount = selectbyquery("prods","Several",TazQuery,)

if tzcount = 1 then do
    rechan = Getfirstrecord(prodvw+"|prods",null)
    setrecordvalues(prodvw,rechan,{{purp,totalintrips}})
end
else
    addrecords(prodvw, {"TAZID",purp}, {{tazid3,totalintrips}},null)

addrecords(tripvw,
{"TAZID", "SZ", "HHS1AO0", "HHS1AO1", "HHS1AO2", "HHS1AO3", "HHS2AO0", "HHS2AO1", "HHS2AO2",
 "HHS2AO3", "HHS3AO0", "HHS3AO1", "HHS3AO2", "HHS3AO3", "HHS4AO0", "HHS4AO1",
"HHS4AO2", "HHS4AO3", "Total", "Int_Total"}, {{tazid3,sz3,trphh1ao0,trphh1ao1,trphh1ao2,trphh1ao3,trphh2ao0,trphh2ao1,trphh
2ao2,
trphh2ao3,trphh3ao0,trphh3ao1,trphh3ao2,trphh3ao3,trphh4ao0,trphh4ao1,trphh4ao2,trphh4ao3,totaltrps,totalintrips}},nul
l)
    trec = GetNextRecord(newdistvw+"|Tony",null,null)
    end
    prec = GetNextRecord(prdvw+"|",null,null)
    end
    closeview(tripvw)
end
closeview(newdistvw)
return(prodvw)
endMacro

// *****End Productions *****
```



```
// ***** This sections does the attraction model *****
macro "attraction_model" (in_value)
  tzvw = in_value[1]
  prodvw = in_value[2]
  thepath = in_value[3]

  tempattvw = runmacro("create_table",{ "tmpattract","tmpattract.dbf","attract",thepath+"\\tg"})
  tazrec = GetFirstRecord(tzvw+"|",null)

  ttlhb watt = 0
  ttlhboatt = 0
  ttlnhbatt = 0
  ttlngatt = 0

  while tazrec <> null do
    ataz = tzvw.("TAZ_ID")
    hb watt = (tzvw.("ZFARMIND") * 1.650857) + (2.26989 * tzvw.("ZGOVT")) + (0.59889 * tzvw.("ZNONIND"))
    hboatt = (tzvw.("HOUSEHOLDS") * 4.878835) + (0.391102 * tzvw.("ZFARMIND")) + (0.197753 * tzvw.("ZNONINDG"))
    nhbatt = (tzvw.("ZFARMIND") * 1.375648) + (1.727858 * tzvw.("ZNONINDG"))
    lngatt = (tzvw.("ZNUMEMP") * 0.407612)
    if hb watt <> null then do
      ttlhb watt = ttlhb watt + hb watt
      ttlhboatt = ttlhboatt + hboatt
      ttlnhbatt = ttlnhbatt + nhbatt
      ttlngatt = ttlngatt + lngatt
    end
    addrecords(tempattvw,{ "TAZID","HBW","NHB","HBO","LNG"},{{ ataz,hb watt,nhbatt,hboatt,lngatt}},null)
    tazrec = GetNextRecord(tzvw+"|",null,null)
  end
  prdrec = GetFirstRecord(prodvw+"|",null)
  ttlhb wprd = 0
  ttlhboprd = 0
  ttlnhbprd = 0
  ttlngprd = 0

  while prdrec <> null do

    hb wprd = prodvw.("HBW")
    hboprd = prodvw.("HBO")
    nhbprd = prodvw.("NHB")
```



```
lngprd = prodvw.("LNG")
ttlhbwprd = hbwprd + ttlhbwprd
ttlhboprd = hboprd + ttlhboprd
ttlnhbprd = nhbprd + ttlnhbprd
ttl lngprd = lngprd + ttl lngprd

prdrec = GetNextRecord(prodvw+"|",null,null)
end
attvw = runmacro("create_table",{ "attract","attract.dbf","attract",thepath+"\\tg"})
attrec = GetFirstRecord(tempattvw+"|",null)

while attrec <> null do
    ataz2 = tempattvw.("TAZID")
    finalhb watt = (tempattvw.("HBW") * ttlhbwprd) / ttlhb watt
    finalhbo watt = (tempattvw.("HBO") * ttlhboprd) / ttlhbo watt
    finalnhb watt = (tempattvw.("NHB") * ttlnhbprd) / ttlnhb watt
    final lng watt = (tempattvw.("LNG") * ttl lngprd) / ttl lng watt

    addrecords(attvw, {"TAZID","HBW","NHB","HBO","LNG"}, {{ataz2,finalhb watt,finalnhb watt,finalhbo watt,final lng watt}},null
)

    attrec = GetNextRecord(tempattvw+"|",null,null)
end
closeview(tempattvw)
ajvw = JoinViews("TAZ+Attract",tzvw+".TAZ_ID",attvw+".TAZID",)
arec = getfirstrecord(ajvw+"|",null)
attrecs = GetRecordCount(ajvw,null)
CreateProgressbar("Updating Attractions...","True")
i=0
while arec <> null do
    i = i+1
    apct = ceil(i / attrecs * 100)
    updateprogressbar("Record #"+String(i),apct)
    ajvw.hbw_attr = ajvw.HBW
    ajvw.hbo_attr = ajvw.HBO
    ajvw.nhb_attr = ajvw.NHB
    ajvw.lng_attr = ajvw.LNG
    arec = GetNextRecord(ajvw+"|",null,null)
end
destroyprogressbar()
```



```
closeview(ajvw)

// put productions values on node layer

pjvw = JoinViews("TAZ+Productions",tzvw+".TAZ_ID",prodvw+".TAZID",)
prec = getfirstrecord(pjvw+"|",null)
CreateProgressbar("Updating Productions...", "True")
i=0
while prec <> null do
    i=i+1
    apct = ceil(i / attrecs * 100)
    updateprogressbar("Record #"+String(i),apct)
    pjvw.hbw_prod = pjvw.HBW
    pjvw.hbo_prod = pjvw.HBO
    pjvw.nhb_prod = pjvw.NHB
    pjvw.lng_prod = pjvw.LNG
    prec = GetNextRecord(pjvw+"|",null,null)
end
destroyprogressbar()
closeview(attvw)
closeview(pjvw)
closeview(prodvw)
deletefile(thepath+"\\tg\\tmpattract.dbf")
deletefile(thepath+"\\tg\\tmpattract.mdx")
endMacro

Macro "balance_ps_to_as" (in_value)

    thepath = in_value[1]
    tazvw = in_value[2]
    tazfilename = in_value[3]
    trec = getfirstrecord(tazvw+"|",null)
    while trec <> null do
        tazvw.nhb_prod = tazvw.nhb_attr
        trec = GetNextRecord(tazvw+"|",null,null)
    end
    RunMacro("TCB Init")
// STEP 1: Balance
    Opts = {"Input", {"Data Set", {tazfilename+"|"+tazvw,
        tazvw,
        "Selection",
```



```
        "Select * where HBW_PROD <> null" }},
    {"Data View",      tazvw}}},
{"Field",      {"Vector 1",      {"["+tazvw+].HBW_PROD",
        {"["+tazvw+].HBO_PROD",
        {"["+tazvw+].NHB_PROD"}},
    {"Vector 2",      {"["+tazvw+].HBW_ATTR",
        {"["+tazvw+].HBO_ATTR",
        {"["+tazvw+].NHB_ATTR"}}}},
{"Global",      {"Pairs",      3},
    {"Holding Method",      {1,
        1,
        1}}},
    {"Percent Weight",      {50,
        50,
        50}},
    {"Sum Weight",      {100,
        100,
        100}},
    {"V1 Options",      {1,
        1,
        1}},
    {"V1 Holding Set",      {,,}},
    {"V2 Options",      {1,
        1,
        1}},
    {"V2 Holding Set",      {,,}}}}

    if !RunMacro("TCB Run Procedure", "Balance", Opts) then Return( RunMacro("TCB Closing", 0) )
endMacro

//*****End Attraction Model *****

//*****End Trip Generation Macros *****

//***** This Section Does Trip Distribution *****

Macro "Multi_Path" (in_value)
    thepath = in_value[1]
    nodes = in_value[2]
    tazvw = in_value[3]
```



```
linefilename = in_value[4]
intzones = string(in_value[5])
pass = string(in_value[6])
thenetwork = in_value[7]
impfilename = "Imp0"+pass+".mtx"
RunMacro("TCB Init")
// STEP 1: TCSPMAT
  Opts = {{"Input",      {{"Origin Set",      {linefilename+"|"+nodes,
                                             nodes,
                                             "Selection",
                                             "Select * where TAZ <= "+intzones}}},
          {"Destination Set", {linefilename+"|"+nodes,
                                             nodes,
                                             "Selection"}}},
          {"Via Set",      {linefilename+"|"+nodes,
                                             nodes}}},
          {"Network",      thenetwork}}},
  {"Field",      {{"Nodes",      "[ "+nodes+" ].ID"}}},
  {"Global",     {{"SP Option",   "Matrix"},
                 {"Minimize",   "FFTIME"},
                 {"Skim Fields", {{"Length",
                                     {"All"},
                                     {"CTIME",
                                     "All"}}}}}},
  {"Output",     {{"Output Matrix", {{"Label",
                                     "Shortest Path"},
                                     {"File Name",
                                     thepath+"\\td\\"+impfilename}}}}}}

if !RunMacro("TCB Run Procedure", "TCSPMAT", Opts) then Return( RunMacro("TCB Closing", 0) )

setview(tazvw)
aquery = "Select * where TAZ_ID <= "+intzones
  numsel = selectbyquery("Tazs","Several",aquery,)
  tazrec = GetFirstRecord(tazvw+"|Tazs",null)
impmat = openmatrix(thepath+"\\td\\"+impfilename,"Auto")
  theSet = nodes+"|Selection"
  newindex =CreateMatrixIndex("TAZID",impmat,"Both",theSet,"ID","TAZ")

while tazrec <> null do
  nodeid = tazvw.ID
```



```
newlen = tazvw.ApproxRadius
newfftime = tazvw.IntraTT
// ctime = tazvw.CTime

lencur = CreateMatrixCurrency(impmat,"Length (Skim)",null,null,)
setmatrixvalue(lencur,string(nodeid),string(nodeid),newlen)
//fillmatrix(lencur,string(nodeid),string(nodeid),{"Copy",newlen},{{"Diagonal",0}})
ffcur = CreateMatrixCurrency(impmat,"FFTIME",null,null,)
setmatrixvalue(ffcur,string(nodeid),string(nodeid),newfftime)
//fillmatrix(ffcur,{string(nodeid)},{string(nodeid)},{ "Copy",newfftime},{ {"Diagonal",0}})
ctcur = CreateMatrixCurrency(impmat,"CTIME (Skim)",null,null,)
setmatrixvalue(ctcur,string(nodeid),string(nodeid),newfftime)
//fillMatrix(ctcur,{string(nodeid)},{string(nodeid)},{ "Copy",newfftime},{ {"Diagonal",0}})
    tazrec = GetNextrecord(tazvw+"|Tazs",null,null)
end
return(impmat)
endMacro

Macro "friction_factors" (in_value)
thepath = in_value[1]
amatrix = in_value[2]
pass = string(in_value[3])

    //openmatrix(thepath+"\\td\\imp01.mtx","Auto")
createTablefromMatrix(amatrix,thepath+"\\td\\tempimp0"+pass+".bin","FFB",{{"complete","yes"}})
avw = opentable("FFactors","FFB",{thepath+"\\td\\ffactors.bin",})
impvw = opentable("TempImp","FFB",{thepath+"\\td\\tempimp0"+pass+".bin",})

rec = getfirstrecord(avw+"|",null)
i = 0
count = 1

while rec <> null do
    i = i+1
    tt = avw.Bins
    ffhbw = avw.HBW
    ffhbo = avw.HBO
    ffnhb = avw.NHB
    fflng = avw.LONG

    if i = 1 then do
```



```
        fffarray = {{ffhbw,ffhbo,ffnhb,fflng}}
    end
    if i > 1 then do
        fffarray = InsertArrayElements(fffarray,i,{{ffhbw,ffhbo,ffnhb,fflng}})
    end
    rec = GetNextRecord(avw+"|",null,null)
end
if (pass = "1") then thefields = {"LNG"}
if pass = "2" then thefields = {"HBW","HBO","NHB"}
atest = RunMacro("addfields",{thefields,impvw})
EnableProgressbar("Status",1)
CreateProgressbar("Calculating Friction Factors...","True")
imprec = getfirstrecord(impvw+"|",null)
acount = getrecordcount(impvw,null)
i = 0

while imprec <> null do
    i = i+1
    apct = ceil(i / acount * 100)
    bol = updateprogressbar("Record #"+String(i),apct)

    ctime = impvw.[CTIME (SKIM)]
    lowctime = Floor(ctime)

    if lowctime > 300 then do
        lowctime = 300
    end

    if lowctime = 0 then
        ctimesubary = fffarray[1]
    else
        ctimesubary = fffarray[lowctime+1]

    hbwff = ctimesubary[1]

    fftime = impvw.FFTIME

    lowfftime = floor(fftime)

    if fftime > 300 then
        lowfftime = 300
    end
end
```



```
if fftime = 0 then
    fffsubary = ffarray[1]
else
    fffsubary = ffarray[lowfftime+1]

hboff = fffsubary[2]
nhbff = fffsubary[3]
lngff = fffsubary[4]
if pass = "1" then setrecordvalues(impvw,null,{{"LNG",lngff}})
if pass = "2" then setrecordvalues(impvw,null,{{"HBW",hbwoff},{ "HBO",hboff},{ "NHB",nhbff}})
imprec = getnextrecord(impvw+"|",null,null)
```

end

destroyprogressbar()

if pass = "1" then do

```
newmatrix = CreateMatrixfromview("Friction Factors",impvw+"|", "RCINDEX", "[RCINDEX:1]", {"LNG"},
    {"File Name",thepath+"\\td\\ffact0"+pass+".mtx"},
    {"Type", "Float"},
    {"Sparse", "No"},
    {"Column Major", "No"},
    {"File Based", "Yes"}})
```

end

if pass = "2" then do

```
newmatrix = CreateMatrixfromview("Friction Factors",impvw+"|", "RCINDEX", "[RCINDEX:1]", {"HBW", "HBO", "NHB"},
    {"File Name",thepath+"\\td\\ffact0"+pass+".mtx"},
    {"Type", "Float"},
    {"Sparse", "No"},
    {"Column Major", "No"},
    {"File Based", "Yes"}})
```

end

```
closeview(impvw)
closeview(avw)
deletefile(thepath+"\\td\\tempimp0"+pass+".bin")
closematrix(newmatrix)
```

endMacro



```
Macro "TripDist" (in_value)
  thepath = in_value[1]
  tazvw = in_value[2]
  tazfilename = in_value[3]
  pass = string(in_value[4])
  intzones = string(in_value[5])

  if (pass = "1") then do
    RunMacro("TCB Init")
    // STEP 1: Balance
    BalOpts = {{"Input",      {"Data Set",      {tazfilename+"|"+tazvw,
                                                tazvw,
                                                "Selection",
                                                "Select * where taz <= "+intzones}}},
              {"Data View",   tazvw}}},
    {"Field",    {"Vector 1",    {"["+tazvw+].LNG_PROD"}},
                 {"Vector 2",    {"["+tazvw+].LNG_ATTR"}}}},
    {"Global",   {"Pairs",      1},
                 {"Holding Method", {1}},
                 {"Percent Weight", {50}},
                 {"Sum Weight",     {100}},
                 {"V1 Options",     {1}},
                 {"V1 Holding Set",  {}},
                 {"V2 Options",     {1}},
                 {"V2 Holding Set",  {}}}}

    if !RunMacro("TCB Run Procedure", "Balance", BalOpts) then Return( RunMacro("TCB Closing", 0) )

    Opts = {{"Input",      {"PA View Set",      {tazfilename+"|"+tazvw,
                                                tazvw,
                                                "Selection",
                                                "Select * where taz <= "+intzones}}},
              {"FF Matrix Currencies",    {"thepath+"\\td\\ffact0"+pass+".mtx",
                                                "LNG",
                                                "RCIndex",
                                                "RCIndex:1"}},
              {"Imp Matrix Currencies",    {"thepath+"\\td\\ffact0"+pass+".mtx",
                                                "LNG",
                                                "RCIndex",
```



```
        "RCIndex:1"}}},
        {"KF Matrix Currencies",    {{thepath+"\\td\\kfactor.mtx",
        "KFACTOR",
        "node",
        "node"}}}}},
{"Field",    {"Prod Fields",    {"["+tazvw+].LNG_PROD"}},
             {"Attr Fields",    {"["+tazvw+].LNG_ATTR"}}}},
{"Global",  {"Purpose Names",   {"LNG"}},
             {"Iterations",     {200}},
             {"Convergence",    {0.01}},
             {"Constraint Type", {"Double"}},
             {"Fric Factor Type", {"Matrix"}},
             {"A List",         {1}},
             {"B List",         {0.3}},
             {"C List",         {0.01}}}},
{"Flag",    {"Use K Factors",    {0}}}},
{"Output",  {"Output Matrix",   {"Label",
        "Output Matrix"},
        {"File Name",
        thepath+"\\td\\ptripdist0"+pass+".mtx"}}}}}}

end
if (pass = "2") then do
    Opts = {"Input",    {"PA View Set",    {tazfilename+"|"+tazvw,
        tazvw,
        "Selection",
        "Select * where HBW_PROD <> null"}},
        {"FF Matrix Currencies",    {{thepath+"\\td\\ffact0"+pass+".mtx",
        "HBW",
        "RCIndex",
        "RCIndex:1"},
        {thepath+"\\td\\ffact0"+pass+".mtx",
        "HBO",
        "RCIndex",
        "RCIndex:1"},
        {thepath+"\\td\\ffact0"+pass+".mtx",
        "NHB",
        "RCIndex",
        "RCIndex:1"}}},
        {"Imp Matrix Currencies",    {{thepath+"\\td\\ffact0"+pass+".mtx",
        "HBW",
        "RCIndex",
```



```

"RCIndex:1"},
{thepath+"\\td\\ffact0"+pass+".mtx",
"HBW",
"RCIndex",
"RCIndex:1"},
{thepath+"\\td\\ffact0"+pass+".mtx",
"HBW",
"RCIndex",
"RCIndex:1"}},
{"KF Matrix Currencies", {{thepath+"\\td\\kfactor.mtx",
"KFACTOR",
"node",
"node"},
{thepath+"\\td\\kfactor.mtx",
"KFACTOR",
"node",
"node"},
{thepath+"\\td\\kfactor.mtx",
"KFACTOR",
"node",
"node"}}}},
{"Field", {{{"Prod Fields", {"["+tazvw+"].HBW_PROD",
["+tazvw+"].HBO_PROD",
["+tazvw+"].NHB_PROD"}},
{"Attr Fields", {"["+tazvw+"].HBW_ATTR",
["+tazvw+"].HBO_ATTR",
["+tazvw+"].NHB_ATTR"}}}}},
{"Global", {{{"Purpose Names", {"HBW",
"HBO",
"NHB"}},
{"Iterations", {200,
200,
200}},
{"Convergence", {0.01,
0.01,
0.01}},
{"Constraint Type", {"Double",
"Double",
"Double"}},
{"Fric Factor Type", {"Matrix",
"Matrix",

```



```
        "Matrix"}},
    {"A List",      {1,
                    1,
                    1}},
    {"B List",      {0.3,
                    0.3,
                    0.3}},
    {"C List",      {0.01,
                    0.01,
                    0.01}}},
    {"Flag",        {"Use K Factors",      {1,
                    1,
                    1}}}},
    {"Output",      {"Output Matrix",      {"Label",
                    "Output Matrix"},
                    {"File Name",
                    thepath+"\\td\\ptripdist0"+pass+".mtx"}}}}}}

end
RunMacro("TCB Init")
if !RunMacro("TCB Run Procedure", "Gravity", Opts) then Return( RunMacro("TCB Closing", 0) )
Return()
endMacro

/*****End Trip Distrbution Macros *****/

/*****No do the mode choice part *****/

Macro "thematrix" (in_value)
    thepath = in_value
    // open impedance matrix
    M = openmatrix(thepath+"\\td\\imp01.mtx", "True")
    // do time matrix here

    mc = CreateMatrixCurrency(M, "FFTIME", null, null, )

    timematrix = copymatrix(mc, {"File Name", thepath+"\\mc\\time.mtx"},
                             {"Label", "Time"},
                             {"File Based", "Yes"})
    dropmatrixcore(timematrix, "Length (Skim)")
```



```
setmatrixcorename(timematrix,"FFTIME","FFLOW*")
setmatrixcorename(timematrix,"CTIME (Skim)","CTIME")
addmatrixcore(timematrix,"FFTRAN_IVTT")
addmatrixcore(timematrix,"FFTRAN_OVTT")
addmatrixcore(timematrix,"CTTRAN_IVTT")
addmatrixcore(timematrix,"CTTRAN_OVTT")

tranmtx = openmatrix(thepath+"\\routes\\TR_skim.mtx","Auto")

ffovttmc = CreateMatrixCurrency(tranmtx,"FFTIME (non-transit)",null,null,)
ctivttmc = CreateMatrixCurrency(tranmtx,"CTIME (in-vehicle)",null,null,)
ctovttmc = CreateMatrixCurrency(tranmtx,"CTIME (non-transit)",null,null,)
ffivttmc = CreateMatrixCurrency(tranmtx,"FFTIME (in-vehicle)",null,null,)

targffivt = CreateMatrixCurrency(timematrix,"FFTRAN_IVTT",null,null,)
MergeMatrixElements(targffivt,{ffivttmc},null,null,{{"Force Missing","Yes"}})
targffovt = CreateMatrixCurrency(timematrix,"FFTRAN_OVTT",null,null,)
MergeMatrixElements(targffovt,{ffovttmc},null,null,{{"Force Missing","Yes"}})
targctivt = CreateMatrixCurrency(timematrix,"CTTRAN_IVTT",null,null,)
MergeMatrixElements(targctivt,{ctivttmc},null,null,{{"Force Missing","Yes"}})
targctovt = CreateMatrixCurrency(timematrix,"CTTRAN_OVTT",null,null,)
MergeMatrixElements(targctovt,{ctovttmc},null,null,{{"Force Missing","Yes"}})
closematrix(timematrix)

// Now do the cost matrix

costmatrix = copymatrix(mc,{{"File Name",thepath+"\\mc\\cost.mtx"},
    {"Label","Cost"},
    {"File Based","Yes"},
    {"Table","Length (Skim)"}})
addmatrixcore(costmatrix,"Auto")
addmatrixcore(costmatrix,"Transit")
addmatrixcore(costmatrix,"Length(Transit)")

lentranc = CreateMatrixCurrency(tranmtx,"Length (in-vehicle)",null,null,)
targlenivt = CreateMatrixCurrency(costmatrix,"Length(Transit)",null,null,)
MergeMatrixElements(targlenivt,{lentranc},null,null,{{"Force Missing","Yes"}})

automc = CreateMatrixCurrency(costmatrix,"Auto",null,null,)
AutoExpr = "30 * [Length (Skim)]"
EvaluateMatrixExpression(automc,AutoExpr,null,null,)
```



```
tranmc = CreateMatrixCurrency(costmatrix, "Transit",null,null,)
TranExpr = "19.50 * [Length(Transit)]"
EvaluateMatrixExpression(tranmc,TranExpr,null,null,)
dropmatrixcore(costmatrix,"Length (Skim)")
endMacro

Macro "mnl_eval_lng" (in_value)
  thepath = in_value[1]
  tazvw = in_value[2]
  tazfilename = in_value[3]
  intzones = string(in_value[4])
  RunMacro("TCB Init")
// STEP 1: MNL Evaluation
  Opts = {{"Input",      {"View Set",      {tazfilename+"|"+tazvw,
                                     tazvw,
                                     "Selection",
                                     "Select * where taz <= "+intzones}}},
          {"Destination Set", {tazfilename+"|"+tazvw,
                               tazvw,
                               "Selection",
                               "Select * where taz <= "+intzones}}},
          {"Model Table",     {thepath+"\\MC\\LNGMODEL.BIN"}},
          {"Matrix Currencies", {thepath+"\\mc\\time.mtx",
                                "FFTRAN_IVTT",
                                "RCIndex",
                                "RCIndex"},
          {thepath+"\\mc\\cost.mtx",
            "Auto",
            "RCIndex",
            "RCIndex"}}}},
          {"Field",          {"ID Field",      "["+tazvw+"].ID"}},
          {"Global",         {"Number of Modes", 2},
          {"Model Name",     "Coefficients"}},
          {"Flag",           {"Aggregate",      1},
          {"Delete Case",    0}}},
          {"Output",         {"Output Matrix",  {"Label",
                                               "Output Matrix"},
          {"File Name",
          thepath+"\\mc\\lngshares.mtx"}}}}}

  if !RunMacro("TCB Run Procedure", "MNL Evaluation", Opts) then Return( RunMacro("TCB Closing", 0) )
```



endMacro

```
Macro "test_matrix" (in_value)
  thepath = in_value
  M = openmatrix(thepath+"\\td\\ptripdist02.mtx","True")
  mchbw = CreateMatrixCurrency(M,"HBW",null,null,)
  hbwmtx = copymatrix(mchbw,{{"File Name",thepath+"\\mc\\hbwbase.mtx"},
    {"Label","HBW Base"},
    {"File Based","Yes"}})
  addmatrixcore(hbwmtx,"Auto")
  addmatrixcore(hbwmtx,"Transit")
  addmatrixcore(hbwmtx,"HBW_VEH_TRP")
  autohbwmc = CreateMatrixCurrency(hbwmtx,"Auto",null,null,)
  hbwexp1 = "[HBW] *0.99"
  EvaluateMatrixExpression(autohbwmc,hbwexp1,null,null,)
  tranhbwmc = CreateMatrixCurrency(hbwmtx,"Transit",null,null,)
  hbwexp2 = "[HBW] * 0.01"
  EvaluateMatrixExpression(tranhbwmc,hbwexp2,null,null,)
  SetMatrixCore(hbwmtx,"Auto")
  hbwaoccmc = CreateMatrixCurrency(hbwmtx,"HBW_VEH_TRP",null,null,)
  hbwaocccexp = "[Auto] / 1.20"
  EvaluateMatrixExpression(hbwaoccmc,hbwaocccexp,null,null,)

  dropmatrixcore(hbwmtx,"HBO")
  dropmatrixcore(hbwmtx,"NHB")
  dropmatrixcore(hbwmtx,"HBW")

  mchbo = CreateMatrixCurrency(M,"hbo",null,null,)
  hbomtx = copymatrix(mchbo,{{"File Name",thepath+"\\mc\\hbobase.mtx"},
    {"Label","HBO Base"},
    {"File Based","Yes"}})
  addmatrixcore(hbomtx,"Auto")
  addmatrixcore(hbomtx,"Transit")
  addmatrixcore(hbomtx,"HBO_VEH_TRP")
  autohbomc = CreateMatrixCurrency(hbomtx,"Auto",null,null,)
  hboexp1 = "[HBO] *0.92"
  EvaluateMatrixExpression(autohbomc,hboexp1,null,null,)
  tranhbomc = CreateMatrixCurrency(hbomtx,"Transit",null,null,)
  hboexp2 = "[HBO] * 0.08"
  EvaluateMatrixExpression(tranhbomc,hboexp2,null,null,)
  hboaoccmc = CreateMatrixCurrency(hbomtx,"HBO_VEH_TRP",null,null,)
```



```
hboaoccecp = "[Auto] / 2.15"
EvaluateMatrixExpression(hboaoccmc,hboaoccecp,null,null,)
SetMatrixCore(hbomtx,"Auto")
dropmatrixcore(hbomtx,"HBO")
dropmatrixcore(hbomtx,"NHB")
dropmatrixcore(hbomtx,"HBW")
mcnhb = CreateMatrixCurrency(M,"nhb",null,null,)
nhbmtx = copymatrix(mcnhb,{{"File Name",thepath+"\\mc\\nhbbase.mtx"},
    {"Label","NHB Base"},
    {"File Based","Yes"}})
addmatrixcore(nhbmtx,"Auto")
addmatrixcore(nhbmtx,"Transit")
addmatrixcore(nhbmtx,"NHB_VEH_TRP")
autonhbmc = CreateMatrixCurrency(nhbmtx,"Auto",null,null,)
nhbexp1 = "[NHB] *0.96"
EvaluateMatrixExpression(autonhbmc,nhbexp1,null,null,)

trannhbmc = CreateMatrixCurrency(nhbmtx,"Transit",null,null,)
nhbexp2 = "[NHB] * 0.04"
EvaluateMatrixExpression(trannhbmc,nhbexp2,null,null,)

nhbaoccmc = CreateMatrixCurrency(nhbmtx,"NHB_VEH_TRP",null,null,)
nhbaoccecp = "[Auto] / 1.87"
EvaluateMatrixExpression(nhbaoccmc,nhbaoccecp,null,null,)

SetMatrixCore(nhbmtx,"Auto")
dropmatrixcore(nhbmtx,"NHB")
dropmatrixcore(nhbmtx,"HBO")
dropmatrixcore(nhbmtx,"HBW")

// ***** Do Long Trips here *****

lngpermat = openmatrix(thepath+"\\td\\ptripdist01.mtx","True")
lngshrmtx = OpenMatrix(thepath+"\\mc\\lngshares.mtx","True")
autolngmc = CreateMatrixCurrency(lngshrmtx,"Auto",null,null,)
lngpermc = CreateMatrixCurrency(lngpermat,"LNG",null,null,)

lngtrpmtx = CombineMatrices({autolngmc,lngpermc}, {{"File Name", thepath+"\\mc\\lngbase.mtx"},
{"Label", "Long Vehicle Trips"},
{"File Based", "Yes"},
{"Operation", "Union"}})
```



```
AddMatrixCore(lngtrpmtx,"LNG_VEH_TRP")

lngbsmc = CreateMatrixCurrency(lngtrpmtx,"LNG_VEH_TRP",null,null,)
lngexp = "([Auto] * [LNG]) / 3.06"
EvaluateMatrixExpression(lngbsmc,lngexp,null,null,)

// ***** Combine All Trips Here *****

totvehtrpmtx = CombineMatrices({hbwaoccmc,hboaoccmc,nhbaoccmc,lngbsmc}, {"File Name",
thepath+"\\mc\\combvehtrp.mtx"}, {"Label", "Vehicle Trips"}, {"File Based", "Yes"}, {"Operation", "Union"}})

AddMatrixCore(totvehtrpmtx,"Total Vehicles")

totmc = CreateMatrixCurrency(totvehtrpmtx,"Total Vehicles",null,null,)

RunMacro("TCB Init")
// STEP 1: Matrix Formula Fill
Opts = {"Input", {"Matrix Currency", {thepath+"\\mc\\combvehtrp.mtx",
"LNG_VEH_TRP",
"Rows",
"Columns"}}}, {"Global", {"Formula", "if [LNG_VEH_TRP] = null then 0 else [LNG_VEH_TRP]}}}

if !RunMacro("TCB Run Operation", "Matrix Formula Fill", Opts) then Return( RunMacro("TCB Closing", 0))

totexp = "[HBW_VEH_TRP]+[HBO_VEH_TRP]+[NHB_VEH_TRP]+[LNG_VEH_TRP]"

EvaluateMatrixExpression(totmc,totexp,null,null,)

transmtx = TransposeMatrix(totvehtrpmtx,{"File Name",thepath+"\\mc\\transpveh.mtx"}, {"Label", "Transposed Vehicles"}, {"Type", "Double"}, {"Sparse", "No"}, {"Column Major", "No"}, {"File Based", "Yes"}})
```



```
AddMatrixCore(totvehtrpmtx, "Half_Vehicles")
AddMatrixCore(transmtx, "Transp_Half_Vehicles")

halfmc = CreateMatrixCurrency(totvehtrpmtx, "Half_Vehicles", null, null, )
halfexp = "[Total Vehicles] / 2"
EvaluateMatrixExpression(halfmc, halfexp, null, null, )

transhalfmc = CreateMatrixCurrency(transmtx, "Transp_Half_Vehicles", null, null, )
transhalfexp = "[Total Vehicles] / 2"
EvaluateMatrixExpression(transhalfmc, transhalfexp, null, null, )

newvehmtx = CombineMatrices({halfmc, transhalfmc}, {"File Name", thepath+"\\mc\\vehicles.mtx"},
{"Label", "New Matrix"},
{"File Based", "Yes"},
{"Operation", "Union"}})

AddMatrixCore(newvehmtx, "Final OD")
finalodmc = CreateMatrixCurrency(newvehmtx, "Final OD", null, null, )
finalexp = "[Half_Vehicles] + [Transp_Half_Vehicles]"
EvaluateMatrixExpression(finalodmc, finalexp, null, null, )

//everything good to here. Should be able to use merge matrix elements

corr18mat = openmatrix(thepath+"\\mc\\corr18_lng.mtx", "True")
corr18mc = CreateMatrixCurrency(corr18mat, "CAR25_LNG", null, null, )

finalmatrix = copymatrix(corr18mc, {"File Name", thepath+"\\mc\\FinalVeh.mtx"},
{"Label", "Final Assignments Vehicles"},
{"File Based", "Yes"}})

AddMatrixCore(finalmatrix, "Final OD")

odmc = CreateMatrixCurrency(finalmatrix, "Final OD", null, null, )
alexp = "[CAR25_LNG] * 0"
EvaluateMatrixExpression(odmc, alexp, null, null, )

AddMatrixCore(finalmatrix, "All Vehicles")
fnmc = CreateMatrixCurrency(finalmatrix, "All Vehicles", null, null, )
aexp = "[CAR25_LNG] * 0"
EvaluateMatrixExpression(fnmc, aexp, null, null, )
corr18mc = CreateMatrixCurrency(corr18mat, "CAR25_LNG", null, null, )
```



```
MergeMatrixElements(odmc, {finalodmc}, null, null, {{"Force Missing", "No"}})
```

```
finalexp = "[CAR25_LNG] + [Final OD]"  
EvaluateMatrixExpression(fnmc, finalexp, null, null, )  
// showmessage("Finished Mode Choice")  
runmacro("closematrices", )
```

endMacro

Macro "matrix\_manip" (in\_value)

```
thepath = in_value
```

```
purplist = {"HBW", "HBO", "NHB", "LNG"}
```

```
// Time of day factor multi dimension list --for {HBW,HBO,NHB,LNG {AM - PM - OP}}
```

```
factlist1 = {{0.3317,0.3025,0.3658},{0.2416,0.2773,0.5080},{0.1206,0.2517,0.6277},{0.0699,0.2373,0.6928}}
```

```
// In this list the PA and AP factors for all purposes PA always comes first
```

```
appalist
```

```
{0.9568,0.0432,0.0913,0.9087,0.5068,0.4932},{0.8993,0.1007,0.3367,0.6633,0.4076,0.5924},{0.5,0.5,0.5,0.5,0.5,0.5},{0.9  
828,0.0172,0.4652,0.5348,0.4632,0.5368}}
```

```
for i = 1 to purplist.length do
```

```
  purp = purplist[i]
```

```
  flist = factlist1[i]
```

```
  appafactlist = appalist[i]
```

```
  hbwmat = openmatrix(thepath+"\\mc\\"+purp+"base.mtx", "True")
```

```
  mc = CreateMatrixCurrency(hbwmat, purp+"_VEH_TRP", null, null, )
```

```
  newmtx = copymatrix(mc, {"File Name", thepath+"\\tod\\"+purp+"fpa.mtx"},
```

```
    {"Label", ""},
```

```
    {"File Based", "Yes"}})
```

```
  addmatrixcore(newmtx, "PA AM Peak")
```

```
  addmatrixcore(newmtx, "PA PM Peak")
```

```
  addmatrixcore(newmtx, "PA Off Peak")
```

```
  mc1 = CreateMatrixCurrency(newmtx, "PA AM Peak", null, null, )
```



```
exp1 = "["+purp+"_VEH_TRP]*"+string(flist[1])+")"
EvaluateMatrixExpression(mc1,exp1,null,null,)

mc2 = CreateMatrixCurrency(newmtx,"PA PM Peak",null,null,)

exp2 = "["+purp+"_VEH_TRP]*"+string(flist[2])+")"
EvaluateMatrixExpression(mc2,exp2,null,null,)

mc3 = CreateMatrixCurrency(newmtx,"PA Off Peak",null,null,)

exp3 = "["+purp+"_VEH_TRP]*"+string(flist[3])+")"
EvaluateMatrixExpression(mc3,exp3,null,null,)

transmtx = TransposeMatrix(newmtx,{{"File Name",thepath+"\\tod\\"+purp+"fap.mtx"},
    {"Label",purp+" AP"},
    {"Type", "Double"},
    {"Sparse", "No"},
    {"Column Major", "No"},
    {"File Based", "Yes"}})

setmatrixcorename(transmtx,"PA AM Peak", "AP AM Peak")
setmatrixcorename(transmtx,"PA PM Peak", "AP PM Peak")
setmatrixcorename(transmtx,"PA OFF Peak", "AP OFF Peak")

paam = CreateMatrixCurrency(newmtx,"PA AM Peak",null,null,)
apam = CreateMatrixCurrency(transmtx,"AP AM Peak",null,null,)
MatrixCellByCell(paam,apam,{{"File Name", thepath+"\\tod\\"+purp+"famod.mtx"},
    {"Label", "AM OD Matrix"},
    {"Type", "Double"},
    {"Sparse", "No"},
    {"Column Major", "No"},
    {"File Based", "Yes"},
    {"Force Missing", "No"},
    {"Operator", 3},
    {"Scale Left", appafactlist[1]},
    {"Scale Right", appafactlist[2]}})

papm = CreateMatrixCurrency(newmtx,"PA PM Peak",null,null,)
appm = CreateMatrixCurrency(transmtx,"AP PM Peak",null,null,)
```



```
MatrixCellByCell(papm, appm, {{"File Name", thepath+"\\tod\\"+purp+"fpmmod.mtx"},
  {"Label", "PM OD Matrix"},
  {"Type", "Double"},
  {"Sparse", "No"},
  {"Column Major", "No"},
  {"File Based", "Yes"},
  {"Force Missing", "No"},
  {"Operator", 3},
  {"Scale Left", appafactlist[3]},
  {"Scale Right", appafactlist[4]}})

paop = CreateMatrixCurrency(newmtx, "PA OFF Peak", null, null, )
apop = CreateMatrixCurrency(transmtx, "AP OFF Peak", null, null, )
MatrixCellByCell(paop, apop, {{"File Name", thepath+"\\tod\\"+purp+"fopod.mtx"},
  {"Label", "OP OD Matrix"},
  {"Type", "Double"},
  {"Sparse", "No"},
  {"Column Major", "No"},
  {"File Based", "Yes"},
  {"Force Missing", "No"},
  {"Operator", 3},
  {"Scale Left", appafactlist[5]},
  {"Scale Right", appafactlist[6]}})
closematrix(newmtx)

end

// Combine AM Peak Matrices here

hbwammtx = openmatrix(thepath+"\\tod\\hbwfamod.mtx", "True")
hboammtx = openmatrix(thepath+"\\tod\\hbofamod.mtx", "True")
lngammtx = openmatrix(thepath+"\\tod\\lngfamod.mtx", "True")
nhbammtx = openmatrix(thepath+"\\tod\\nhbfamod.mtx", "True")

setmatrixcorename(hbwammtx, "Matrix 1", "HBW AM Peak")
setmatrixcorename(hboammtx, "Matrix 1", "HBO AM Peak")
setmatrixcorename(lngammtx, "Matrix 1", "LNG AM Peak")
setmatrixcorename(nhbammtx, "Matrix 1", "NHB AM Peak")

hbwammc = CreateMatrixCurrency(hbwammtx, "HBW AM Peak", null, null, )
hboammc = CreateMatrixCurrency(hboammtx, "HBO AM Peak", null, null, )
lngammc = CreateMatrixCurrency(lngammtx, "LNG AM Peak", null, null, )
```



```
nhbammc = CreateMatrixCurrency(nhbammtx,"NHB AM Peak",null,null,)

ammtx = CombineMatrices({hbwammc, hboammc,nhbammc,lngammc}, {"File Name", thepath+"\\tod\\famod.mtx"},
{"Label", "AM Peak OD"},
{"File Based", "Yes"},
{"Operation", "Union"})

RunMacro("TCB Init")
// STEP 1: Matrix Formula Fill
  Opts = {"Input", {"Matrix Currency", {thepath+"\\tod\\famod.mtx",
  "LNG AM Peak",
  "Rows",
  "Columns"}}},
  {"Global", {"Formula", "if [LNG AM Peak] = null then 0 else [LNG AM Peak]}}}

if !RunMacro("TCB Run Operation", "Matrix Formula Fill", Opts) then Return( RunMacro("TCB Closing", 0))

ammc = CreateMatrixCurrency(ammtx,"HBW AM Peak",null,null,)
amexp = "[HBW AM Peak]+[HBO AM Peak]+[LNG AM Peak]+[NHB AM Peak]"
evaluatematrixexpression(ammc,amexp,null,null,)
setmatrixcorename(ammtx,"HBW AM Peak","AM Peak")
ammc = CreateMatrixCurrency(ammtx,"AM Peak",null,null,)

dropmatrixcore(ammtx,"HBO AM Peak")
dropmatrixcore(ammtx,"NHB AM Peak")
dropmatrixcore(ammtx,"LNG AM Peak")

closematrix(hbwammtx)
closematrix(hboammtx)
closematrix(lngammtx)
closematrix(nhbammtx)

hbwpmmtx = openmatrix(thepath+"\\tod\\hbwfpmod.mtx","True")
hbopmmtx = openmatrix(thepath+"\\tod\\hbopfmod.mtx","True")
lngpmmtx = openmatrix(thepath+"\\tod\\lngfpmod.mtx","True")
nhbpmmtx = openmatrix(thepath+"\\tod\\nhbfpmod.mtx","True")

setmatrixcorename(hbwpmmtx,"Matrix 1", "HBW PM Peak")
setmatrixcorename(hbopmmtx,"Matrix 1", "HBO PM Peak")
```



```
setmatrixcorename(lngpmmtx,"Matrix 1", "LNG PM Peak")
setmatrixcorename(nhbpmmtx,"Matrix 1", "NHB PM Peak")

hbwpmmc = CreateMatrixCurrency(hbwpmmtx,"HBW PM Peak",null,null,)
hbopmmc = CreateMatrixCurrency(hbopmmtx,"HBO PM Peak",null,null,)
lngpmmc = CreateMatrixCurrency(lngpmmtx,"LNG PM Peak",null,null,)
nhbpmmmc = CreateMatrixCurrency(nhbpmmtx,"NHB PM Peak",null,null,)

pmmtx = CombineMatrices({hbwpmmc, hbopmmc,nhbpmmmc,lngpmmc}, {"File Name", thepath+"\\tod\\fpmmod.mtx"},
{"Label", "PM Peak OD"},
{"File Based", "Yes"},
{"Operation", "Union"}})

RunMacro("TCB Init")
// STEP 1: Matrix Formula Fill
  Opts = {"Input",      {"Matrix Currency",      {thepath+"\\tod\\fpmmod.mtx",
                                                "LNG PM Peak",
                                                "Rows",
                                                "Columns"}}}},
        {"Global",    {"Formula",              "if [LNG PM Peak] = null then 0 else [LNG PM Peak]"}}}}
if !RunMacro("TCB Run Operation", "Matrix Formula Fill", Opts) then Return( RunMacro("TCB Closing", 0))

pmmc = CreateMatrixCurrency(pmmtx,"HBW PM Peak",null,null,)
pmexp = "[HBW PM Peak]+[HBO PM Peak]+[LNG PM Peak]+[NHB PM Peak]"
evaluatematrixexpression(pmmc,pmexp,null,null,)
setmatrixcorename(pmmtx,"HBW PM Peak","PM Peak")
pmmc = CreateMatrixCurrency(pmmtx,"PM Peak",null,null,)

dropmatrixcore(pmmtx,"HBO PM Peak")
dropmatrixcore(pmmtx,"NHB PM Peak")
dropmatrixcore(pmmtx,"LNG PM Peak")

closematrix(hbwpmmtx)
closematrix(hbopmmtx)
closematrix(lngpmmtx)
closematrix(nhbpmmtx)

hbwopmtx = openmatrix(thepath+"\\tod\\hbwfopod.mtx","True")
hbopmtx = openmatrix(thepath+"\\tod\\hbopod.mtx","True")
```



```
lngopmtx = openmatrix(thepath+"\\tod\\lngfopod.mtx","True")
nhbopmtx = openmatrix(thepath+"\\tod\\nhbfopod.mtx","True")

setmatrixcorename(hbwopmtx,"Matrix 1", "HBW OFF Peak")
setmatrixcorename(hboopmtx,"Matrix 1", "HBO OFF Peak")
setmatrixcorename(lngopmtx,"Matrix 1", "LNG OFF Peak")
setmatrixcorename(nhbopmtx,"Matrix 1", "NHB OFF Peak")

hbwoPMC = CreateMatrixCurrency(hbwopmtx,"HBW OFF Peak",null,null,)
hboopmc = CreateMatrixCurrency(hboopmtx,"HBO OFF Peak",null,null,)
lngopmc = CreateMatrixCurrency(lngopmtx,"LNG OFF Peak",null,null,)
nhbopmc = CreateMatrixCurrency(nhbopmtx,"NHB OFF Peak",null,null,)

opmtx = CombineMatrices({hbwoPMC,hboopmc,nhbopmc,lngopmc}, {"File Name", thepath+"\\tod\\fopod.mtx"},
{"Label", "OFF Peak OD"},
{"File Based", "Yes"},
{"Operation", "Union"})

RunMacro("TCB Init")
// STEP 1: Matrix Formula Fill
Opts = {"Input", {"Matrix Currency", {thepath+"\\tod\\fopod.mtx",
"LNG OFF Peak",
"Rows",
"Columns"}},},
{"Global", {"Formula", "if [LNG OFF Peak] = null then 0 else [LNG OFF Peak]}}}

if !RunMacro("TCB Run Operation", "Matrix Formula Fill", Opts) then Return( RunMacro("TCB Closing", 0))

opmc = CreateMatrixCurrency(opmtx,"HBW OFF Peak",null,null,)
opexp = "[HBW OFF Peak]+[HBO OFF Peak]+[LNG OFF Peak]+[NHB OFF Peak]"
evaluatematrixexpression(opmc,opexp,null,null,)
setmatrixcorename(opmtx,"HBW OFF Peak","OFF Peak")
opmc = CreateMatrixCurrency(opmtx,"OFF Peak",null,null,)

dropmatrixcore(opmtx,"HBO OFF Peak")
dropmatrixcore(opmtx,"NHB OFF Peak")
dropmatrixcore(opmtx,"LNG OFF Peak")

// Do Corr18 stuff here
```



```
amtx = openmatrix(thepath+"\\mc\\corr18_lng.mtx","True")
amc = CreateMatrixCurrency(amtx,"CAR25_LNG",null,null,)
corr18mtx = copymatrix(amc,{"File Name",thepath+"\\tod\\todcorr18.mtx"},
    {"Label","Corridor 18 External"},
    {"File Based","Yes"}})

addmatrixcore(corr18mtx,"Corr18 AM Peak")
addmatrixcore(corr18mtx,"Corr18 PM Peak")
addmatrixcore(corr18mtx,"Corr18 OFF Peak")

cammc = CreateMatrixCurrency(corr18mtx,"Corr18 AM Peak",null,null,)
camexp = "[CAR25_LNG] * 0.2063"
EvaluateMatrixExpression(cammc,camexp,null,null,)

cpmmc = CreateMatrixCurrency(corr18mtx,"Corr18 PM Peak",null,null,)
cpmexp = "[CAR25_LNG] * 0.2373"
EvaluateMatrixExpression(cpmmc,cpmexp,null,null,)

copmc = CreateMatrixCurrency(corr18mtx,"Corr18 OFF Peak",null,null,)
copexp = "[CAR25_LNG] * 0.5564"
EvaluateMatrixExpression(copmc,copexp,null,null,)

aexp = "[CAR25_LNG] * 0"
addmatrixcore(corr18mtx,"AM Peak")
theammc = CreateMatrixCurrency(corr18mtx,"AM Peak",null,null,)
EvaluateMatrixExpression(theammc,aexp,null,null,)
addmatrixcore(corr18mtx,"PM Peak")
thepmmc = CreateMatrixCurrency(corr18mtx,"PM Peak",null,null,)
EvaluateMatrixExpression(thepmmc,aexp,null,null,)
addmatrixcore(corr18mtx,"OFF Peak")
theopmc = CreateMatrixCurrency(corr18mtx,"OFF Peak",null,null,)
EvaluateMatrixExpression(theopmc,aexp,null,null,)

MergeMatrixElements(theammc,{ammc},null,null,{"Force Missing","No"})
MergeMatrixElements(thepmmc,{pmmc},null,null,{"Force Missing","No"})
MergeMatrixElements(theopmc,{opmc},null,null,{"Force Missing","No"})

amexp = "[Corr18 AM Peak] + [AM Peak]"
addmatrixcore(corr18mtx,"Final AM Peak")
thefammc = CreateMatrixCurrency(corr18mtx,"Final AM Peak",null,null,)
EvaluateMatrixExpression(thefammc,amexp,null,null,)
```



```
pmexp = "[Corr18 PM Peak] + [PM Peak]"
addmatrixcore(corr18mtx,"Final PM Peak")
thefpmmc = CreateMatrixCurrency(corr18mtx,"Final PM Peak",null,null,)
EvaluateMatrixExpression(thefpmmc,pmexp,null,null,)

opexp = "[Corr18 OFF Peak] + [OFF Peak]"
addmatrixcore(corr18mtx,"Final OFF Peak")
thefopmc = CreateMatrixCurrency(corr18mtx,"Final OFF Peak",null,null,)
EvaluateMatrixExpression(thefopmc,opexp,null,null,)

// trucks

atmtx = openmatrix(thepath+"\\hwy\\CORR18_25_TRK.MTX","True")
tmc = CreateMatrixCurrency(atmtx,"TRUCK2025",null,null,)

trkmtx = copymatrix(tmc,{{"File Name",thepath+"\\hwy\\todtruck.mtx"},
                        {"Label","Truck TOD"},
                        {"File Based","Yes"}})

addmatrixcore(trkmtx,"AM Peak")
addmatrixcore(trkmtx,"PM Peak")
addmatrixcore(trkmtx,"OFF Peak")

ammc = CreateMatrixCurrency(trkmtx,"AM Peak",null,null,)
amexp = "[TRUCK2025] * 0.1714"
EvaluateMatrixExpression(ammc,amexp,null,null,)

pmmc = CreateMatrixCurrency(trkmtx,"PM Peak",null,null,)
pmexp = "[TRUCK2025] * 0.1655"
EvaluateMatrixExpression(pmmc,pmexp,null,null,)

opmc = CreateMatrixCurrency(trkmtx,"OFF Peak",null,null,)
opexp = "[TRUCK2025] * 0.6633"
EvaluateMatrixExpression(opmc,opexp,null,null,)
runmacro("closematrices",)

endMacro

macro "truckassign" (in_value)
  thepath = in_value[1]
  linefilename = in_value[2]
```



```
linevw = in_value[3]
thenetwork = in_value[4]
trkcore = in_value[5]
if trkcore = "OFF Peak" then capfield = "AB_CAP" else capfield = "AB_CAPPEAK"

afilename = thepath+"\\hwy\\truck"+trkcore+"_asn_link.bin"
if trkcore = "AM Peak" then thefields = {"AM_TRK_VOL"}
  if trkcore = "PM Peak" then thefields = {"PM_TRK_VOL"}
  if trkcore = "OFF Peak" then thefields = {"OP_TRK_VOL"}
  RunMacro("addfields",{thefields,linevw})
RunMacro("TCB Init")
// STEP 1: Assignment
Opts = {"Input",      {"Database",      linefilename},
        {"Network",   thenetwork},
        {"OD Matrix Currency", {thepath+"\\hwy\\todtruck.MTX",
                                trkcore,
                                "NodeID",
                                "NodeID"}}}},
        {"Field",     {"FF Time",      "FFTIME"},
        {"Capacity",  capfield},
        {"Alpha",     "None"},
        {"Beta",      "None"}}},
        {"Global",    {"Load Method",   5},
        {"Alpha Value", 0.15},
        {"Beta Value",  4},
        {"Iterations", 20},
        {"Convergence", 0.01}}},
        {"Flag",      {"Do Emission",  0},
        {"Do Tabulation", 0}}},
        {"Output",    {"Flow Table",   afilename}}}}

if !RunMacro("TCB Run Procedure", "Assignment", Opts) then Return(RunMacro("TCB Closing", 0))

asgvw = Opentable("truck_asgn","FFB",{afilename,})
jnvw = JoinViews(linevw+asgvw,linevw+".ID",asgvw+".ID1",)

arec = GetFirstRecord(jnvw+"|",null)
while (arec <> null) do
  jnvw.(thefields[1]) = jnvw.Tot_Flow
  arec = GetNextRecord(jnvw+"|",null,null)
end
```



```
        closeview(asgvw)
        return(jnvw)
endMacro
Macro "assign" (in_value)
    thepath = in_value[1]
    linefilename = in_value[2]
    linevw = in_value[3]
    thenetwork = in_value[4]
    todcore = in_value[5]
    afilename = thepath+"\\hwy\\"+todcore+"_asn_link.bin"
    if todcore = "Final OFF Peak" then capfield = "AB_CAP" else capfield = "AB_CAPPEAK"

    if todcore = "Final AM Peak" then thefields = {"AM_AUTO_VOL"}
        if todcore = "Final PM Peak" then thefields = {"PM_AUTO_VOL"}
            if todcore = "Final OFF Peak" then thefields = {"OP_AUTO_VOL"}
                RunMacro("addfields",{thefields,linevw})
            RunMacro("TCB Init")
        // STEP 1: Assignment
        Opts = {{"Input",          {{"Database",          linefilename},
                                {"Network",            thenetwork},
                                {"OD Matrix Currency",  {thepath+"\\tod\\todcorr18.mtx",
                                                         todcore,
                                                         "From",
                                                         "To"}}}},
              {"Field",         {{"FF Time",          "FFTIME"},
                                {"Capacity",         capfield},
                                {"Alpha",            "None"},
                                {"Beta",             "None"},
                                {"Preload",          "AB_FLOW"}}}},
              {"Global",        {{"Load Method",      5},
                                {"Alpha Value",     0.15},
                                {"Beta Value",      4},
                                {"Iterations",      30},
                                {"Convergence",     0.01}}}},
              {"Flag",          {{"Do Emission",     0},
                                {"Do Tabulation",    0}}}},
              {"Output",        {{"Flow Table",      afilename}}}}

        if !RunMacro("TCB Run Procedure", "Assignment", Opts) then Return( RunMacro("TCB Closing", 0) )

        asgvw = Opentable("auto_asgn","FFB",{afilename,})
```



```
jnvw = JoinViews(linevw+asgvw,linevw+".ID",asgvw+".ID1",)

  arec = GetFirstRecord(jnvw+"|",null)
  while (arec <> null) do
    jnvw.(thefields[1]) = jnvw.Tot_Flow
    arec = GetNextRecord(jnvw+"|",null,null)
  end
  closeview(asgvw)
  closeview(jnvw)

endMacro

macro "closematrices"
  matrixlist = GetMatrices()
  if (matrixlist.length > 0) then do
    allmatrices = matrixlist[1]
    for m = 1 to allmatrices.length do
      closematrix(allmatrices[m])
    end
  end
endMacro

macro "finalcalcs" (in_value)
  linevw = in_value[1]
  thefields = {"Dly_Auto_Vol","Dly_Trk_Vol","Dly_Tot_Vol"}
  runmacro("addfields",{thefields,linevw})
  arec = GetFirstRecord(linevw+"|",null)
  while (arec <> null) do
    linevw.Dly_Auto_Vol = linevw.AM_Auto_Vol + linevw.PM_Auto_Vol + linevw.OP_Auto_Vol
    linevw.Dly_Trk_Vol = linevw.AM_Trk_vol + linevw.PM_Trk_vol + linevw.OP_Trk_Vol
    linevw.Dly_Tot_Vol = linevw.Dly_Auto_Vol + linevw.Dly_Trk_Vol
    arec = GetNextRecord(linevw+"|",null,null)
  end
endMacro

//*****Below are some utility macros *****

macro "create_table" (invalue)
```



// This macro creates a bunch of tables based on the need. Three values  
// are passed to this macro, a view name, a file name ( no path ), and a type of table

```
atest = invalue[3]
if atest = "trip" then do
  CreateTable(invalue[1],invalue[4]+"\\ "+invalue[2],"DBASE",{
    {"TAZID","Integer",5,0,"Yes"},
    {"SZ","Integer",5,0,"No"},
    {"HHS1AO0","Real",10,1,"No"},
    {"HHS1AO1","Real",10,1,"No"},
    {"HHS1AO2","Real",10,1,"No"},
    {"HHS1AO3","Real",10,1,"No"},
    {"HHS2AO0","Real",10,1,"No"},
    {"HHS2AO1","Real",10,1,"No"},
    {"HHS2AO2","Real",10,1,"No"},
    {"HHS2AO3","Real",10,1,"No"},
    {"HHS3AO0","Real",10,1,"No"},
    {"HHS3AO1","Real",10,1,"No"},
    {"HHS3AO2","Real",10,1,"No"},
    {"HHS3AO3","Real",10,1,"No"},
    {"HHS4AO0","Real",10,1,"No"},
    {"HHS4AO1","Real",10,1,"No"},
    {"HHS4AO2","Real",10,1,"No"},
    {"HHS4AO3","Real",10,1,"No"},
    {"Total","Real",10,0,"No"},
    {"Int_Total","Real",10,0,"No"}}})
end
if atest = "hhdist" then do
  CreateTable(invalue[1],invalue[4]+"\\ "+invalue[2],"DBASE",{
    {"TAZID","Integer",5,0,"Yes"},
    {"SZ","Integer",5,0,"No"},
    {"HHS1AO0","Real",10,1,"No"},
    {"HHS1AO1","Real",10,1,"No"},
    {"HHS1AO2","Real",10,1,"No"},
    {"HHS1AO3","Real",10,1,"No"},
    {"HHS2AO0","Real",10,1,"No"},
    {"HHS2AO1","Real",10,1,"No"},
    {"HHS2AO2","Real",10,1,"No"},
    {"HHS2AO3","Real",10,1,"No"},
    {"HHS3AO0","Real",10,1,"No"},
    {"HHS3AO1","Real",10,1,"No"},
```



```
    {"HHS3AO2", "Real", 10, 1, "No"},  
    {"HHS3AO3", "Real", 10, 1, "No"},  
    {"HHS4AO0", "Real", 10, 1, "No"},  
    {"HHS4AO1", "Real", 10, 1, "No"},  
    {"HHS4AO2", "Real", 10, 1, "No"},  
    {"HHS4AO3", "Real", 10, 1, "No"}))
```

end

if atest = "prod" then do

```
    CreateTable(invalue[1], invalue[4]+"\\"+invalue[2], "DBASE", {  
        {"TAZID", "Integer", 5, 0, "Yes"},  
        {"HBW", "Integer", 10, 0, "No"},  
        {"NHB", "Integer", 10, 0, "No"},  
        {"HBO", "Integer", 10, 0, "No"},  
        {"LNG", "Integer", 10, 0, "No"}})
```

end

if atest = "attract" then do

```
    CreateTable(invalue[1], invalue[4]+"\\"+invalue[2], "DBASE", {  
        {"TAZID", "Integer", 5, 0, "Yes"},  
        {"HBW", "Integer", 10, 0, "No"},  
        {"NHB", "Integer", 10, 0, "No"},  
        {"HBO", "Integer", 10, 0, "No"},  
        {"LNG", "Integer", 10, 0, "No"}})
```

end

hh = OpenTable(invalue[1], "DBASE", {invalue[4]+"\\"+invalue[2],})

return (hh)

endMacro

macro "addfields" (in\_value)

fldnames = in\_value[1]

struct = GetTableStructure(in\_value[2])

viewflds = getFields(in\_value[2], numeric)

for i=1 to struct.length do

```
    struct[i]=struct[i]+{struct[i][1]}
```

end

for i=1 to fldnames.length do

```
    pos = ArrayPosition(viewflds[1], {fldnames[i]},)
```

```
    if pos = 0 then do
```



```
        newstr = newstr + {{fldnames[i],"Real", 10, 3,"false",null,null,null,null}}
modtab = 1
end
end

if modtab = 1 then do
    newstr = struct+newstr
    ModifyTable(in_value[2],newstr)
end

endMacro

// *****Create Transit Routes, network and Skims.  Not Currently Used -- Caliper Needs to Debug
// *****8
//  rtes = RunMacro("createroutes",{thepath,cnet,linefilename,linevw})
//  RunMacro("buffers",{linevw,nodevw})
//  tranet = RunMacro("transitnetwork",{thepath,linevw})
//  runmacro("transkim")
Macro "createroutes"
    RunMacro("TCB Init")
// STEP 1: Create RS From Table
    Opts = {{ "Input",      {{ "Network",      "D:\\PROJECTS\\6956\\1998BASE\\HWY\\HIGHWAY_INTNL.NET"},
                          {{ "Link Set",      "D:\\Projects\\6956\\1998Base\\hwy\\I69_base_2000.DBD|Base_2000
Links",
                          {{ "Base_2000 Links"}}},
                          {{ "Tour Table",      {"D:\\PROJECTS\\6956\\STOPTABLE.DBF"}}}},
    {"Global",      {{ "Cost Field",      1},
                    {{ "Route ID Field",    3},
                    {{ "Node ID Field",     4},
                    {{ "Include Stop",      1},
                    {{ "Stop Flag Field",    2},
                    {{ "User ID Field",     4}}}},
    {"Output",      {{ "Output Routes",    "D:\\Projects\\6956\\1998Base\\routes\\test.rts"}}}}
if !RunMacro("TCB Run Operation", "Create RS From Table", Opts) then Return( RunMacro("TCB Closing", 0) )
else
    RunMacro("TCB Closing", 1)
endMacro

Macro "buffers"
    setlayer("Base_2000 Links")
```



```
linkselect =SelectByVicinity("17mirad", "Several", "Stops|", 15.0)
//showmessage(string(linkselect))
setlayer("Base_2000 Node")
tazselect = SelectByVicinity("Selection", "Several", "Stops|", 13.0)
aQuery = "Select * where taz < 743"
SelectByQuery("Selection","Subset",aQuery,)
endMacro

Macro "transitnetwork"
rfllds = {"Route_id","Vehicle Routes.Route_id"}

lfllds = {"Length","Base_2000 Links.Length"},
         {"FFtime","Base_2000 Links.FFTime"},
         {"Ctime","Base_2000 Links.CTime"}}

dim llflds[lfllds.length]
for j = 1 to lfllds.length do
    llflds[j] = lfllds[j]+{"SUMFRAC"}
end

opts = {"Route Attributes", rfllds},
       {"Street Attributes", lfllds},
       {"Walk","Yes"},
       {"Snap Dist",0.05},
       {"Link Direction Field","Base_2000 Links.Dir"},
       {"Link Attributes", llflds},
       {"Snap Stops", "Yes"},
       {"Snap Stop Distance", 0.05}}

showarray(rfllds)
showarray(lfllds)
showarray(opts)
curversion = RunMacro("TC30 Get Transit Network Version")
setlayer("Vehicle Routes")
thenetwork = createTransitNetwork("Stops","Base_2000
Links|17mirad","d:\\projects\\6956\\1998base\\routes\\98tran.tnw",,opts)
SetNetworkInformationItem(thenetwork,"Version",{curversion})
runmacro("transkim")
endMacro

Macro "transkim"
    RunMacro("TCB Init")
```



```
// STEP 1: Transit Skim
  Opts = {{"Input",      {{"Database",      "D:\\Projects\\6956\\1998Base\\hwy\\I69_base_2000.DBD"},
                        {"Network",       "D:\\Projects\\6956\\1998Base\\routes\\98tran.tnw"},
                        {"Origin Set",     {"D:\\Projects\\6956\\1998Base\\hwy\\I69_base_2000.DBD|Base_2000
Node",
                        "Base_2000 Node",
                        "Selection"}}},
                        {"Destination Set", {"D:\\Projects\\6956\\1998Base\\hwy\\I69_base_2000.DBD|Base_2000
Node",
                        "Base_2000 Node",
                        "Selection"}}}},
  {"Global",  {{"SP Method",  1},
               {"Skim Var",   {6,
                               7,
                               8}}},
               {"OD Layer Type", 2}}},
  {"Output",  {{"Skim Matrix", {"Label",
                               "Skim Matrix (Shortest Path)",
                               {"File Name",
                               "D:\\Projects\\6956\\1998Base\\routes\\TR_SKIM5.MTX"}}}}}}

  if !RunMacro("TCB Run Procedure", "Transit Skim", Opts) then goto quit
done:
Return( RunMacro("TCB Closing", 1) )
quit:
Return( RunMacro("TCB Closing", 0) )
endMacro
```