Metropolitan Transportation Planning Organization
for the Gainesville Urbanized Area
Gainesville Urbanized Area Transportation Study

Year 2045 Long-Range Transportation Plan Update

Technical Report 4: 2015 Model Update and Validation

Prepared by:
THE CORRADINO GROUP

Metropolitan Transportation Planning Organization

For the Gainesville Urbanized Area

YEAR 2045 LONG-RANGE TRANSPORTATION PLAN UPDATE

Technical Report 4

2015 Model Update and Validation

The preparation of this report has been financed in part through grants from the Federal Highway Administration, Federal Transit Administration and United States Department of Transportation, under the State Planning and Research Program, Section 505 (or Metropolitan Planning Program, Section 104 (f) of Title 23, United States Code. The contents of this report do not necessarily reflect the official views or policy of the United States Department of Transportation

TABLE of CONTENTS

APPENDICES

LIST of TABLES

LIST of FIGURES

1. INTRODUCTION

This report documents the update and validation of the Gainesville Urbanized Area Transportation Study model for the 2015 base year. The travel demand model was conducted as an integral part of the Gainesville Urbanized Area Year 2045 Long-Range Transportation Plan update. The 2015 model study area covers all of Alachua County, including the nine municipalities within the County (Cities of Alachua, Archer, Gainesville, Hawthorne, High Springs, Newberry, Waldo, and Towns of La Crosse and Micanopy). The effort involved updating the input files of the previous 2010 model to 2015, reviewing and enhancing the model parameters and scripts for trip generation, distribution, highway, and transit network development, mode choice, assignment, and reporting of steps. The Florida Department of Transportation's Florida Standard Urban Transportation Model Structure Cube Framework Phase II: Model Calibration and Validation Standards were followed as a technical guide for the validation effort.

The structure of the report follows the sequence of the processes as in the model chain. It starts with documenting the update of the external trips in Section 2. Sections 3 and 4 cover trip generation and trip distribution. Section 5 documents transit accessibility and path building followed by mode choice in Section 6. Sections 7 and 8 document highway and transit assignments. Finally, Section 9 concludes the report with closing remarks.

2. EXTERNAL TRIPS

The external zones are placed along roadways entering and leaving the Gainesville Urbanized Area Transportation Model Study Area. The zones used in the 2015 base model were reviewed and updated based on the 2010 model. There are 26 external zones outside of the study area, and they are assigned unique Traffic Analysis Zone numbers from 700 to 725.

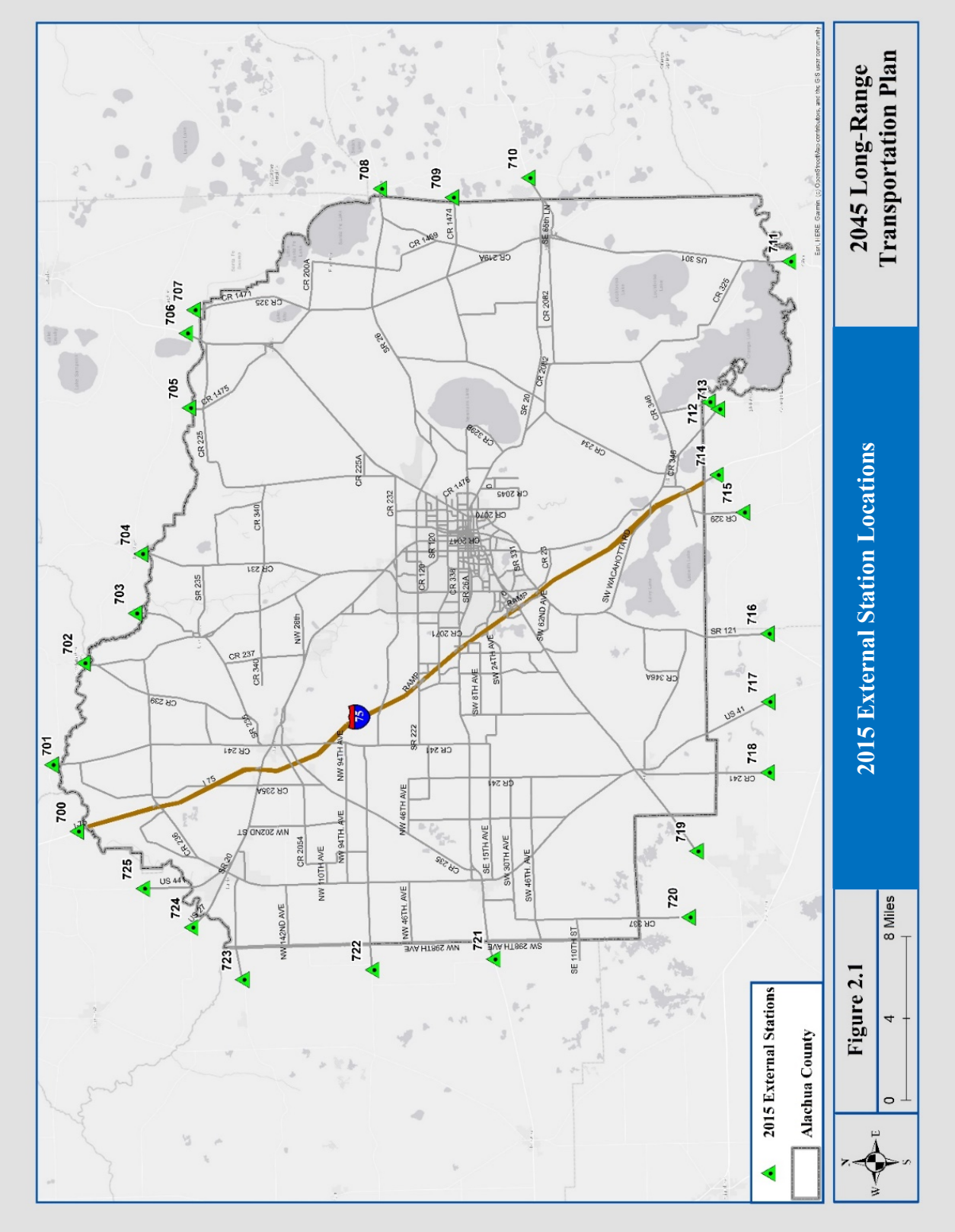
2.1 Internal-External and External-External Trips

External trips are vehicle trips having at least one trip end outside of the model study area. External trips include two categories: Internal-to-External trips or External-to-Internal trips and External-to-External trips. External-External trips are trips with each trip ends outside the study area, and trips with one end inside and one end outside of the study area are called either Internal-External trips or External-Internal trips depending on the origin and destination.

External trips are generated or attracted at the external stations located outside of the model boundary. As described above, these stations are points of entry or exit of the study area. The external stations for the Gainesville Urbanized Area Transportation Study county-wide travel demand model are shown in Figure 2.1.

In the validation process of external trips, the External-External and External-Internal should ideally be estimated based on an external station origin-destination travel survey. In the absence of a recent origin-destination travel survey, the base year 2015 External-External and External-Internal trips were derived from the 2010 External-External and External-Internal trips factored by the 2015 traffic counts obtained from Florida Department of Transportation and Alachua County. Two input files influence external trips named, EETARGET and EITRIPS with scenario year as the suffix. The percent of External-External and External-Internal trips were calculated from the 2010 model. Although the external trips are derived from traffics counts without an origin-destination travel survey, it is necessary to adjust based on local and logical knowledge. These adjustments were conducted iteratively to ensure that generated traffic volumes correspond to the 2015-year traffic counts.

Figure 2.1 2015 External Station Locations



The final 2015 External-External and External-Internal trips are shown in Table 2.1. Since the traffic patterns in Florida are heavily impacted by seasonality, the annual average daily traffic count is converted to peak-season weekday average daily traffic count via model output conversion factor before coding in the model. The percentage of External-External and External-Internal trips are applied to the peak-season weekday average daily traffic for each external zone to calculate total External-External and External-Internal trips.

Table 2.1 Internal-External and External-Internal Percentage Splits

External Traffic Analysis Zone	Total External Trips	Roadways	Total IE Trips	Total EE Trips	IE Percent	EE Percent
700	48,404	I-75 (North) at Columbia County Line	6,800	41,604	14%	86%
701	2,042	County Road 241 (North) at Union County Line	1,388	654	68%	32%
702	5,510	State Road 121 (North) at Union County Line	3,802	1,708	69%	31%
703	154	County Road 237 (North) at Bradford County Line	132	22	86%	14%
704	3,062	SR 235 (North) at Bradford County Line	2,786	276	91%	9%
705	670	County Road 1475 (North) at Bradford County Line	468	202	70%	30%
706	24,373	US 301 (North) at Bradford County Line	9,505	14,868	39%	61%
707	1,133	County Road 325 (North) at Bradford County Line	781	352	69%	31%
708	6,431	State Road 26 (East) at Putnam County Line	3,215	3,216	50%	50%
709	412	County Road 1474 (East) at Putnam County Line	264	148	64%	36%
710	8,831	State Road 20 (East) at Putnam County Line	4,415	4,416	50%	50%
711	12,755	US 301 (North) at Marion County Line	1,403	11,352	11%	89%
712	351	County Road 225 (South) at Marion County Line	295	56	84%	16%
713	8,163	US 441 (South) at Marion County Line	7,265	898	89%	11%
714	53,646	Interstate 75 (South) at Marion County Line	10,000	43,646	19%	81%
715	1,372	County Road 234 (South) at Marion County Line	892	480	65%	35%
716	6,907	State Road 121 (South) at Levy County Line	5,249	1,658	76%	24%
717	4,521	State Road 45 (South) at Levy County Line	3,209	1,312	71%	29%
718	1,078	County Road 241 (South) at Levy County Line	830	248	77%	23%
719	7,448	State Road 24 (Southwest) at Levy County Line	5,288	2,160	71%	29%
720	1,234	County Road 337 (South) at Levy County Line	888	346	72%	28%
721	10,781	State Road 26 (West) at Gilchrist County Line	8,193	2,588	76%	24%
722	2,058	County Road 232 (West) at Gilchrist County Line	1,482	576	72%	28%
723	5,096	NW 182 (West) at Gilchrist County Line	3,618	1,478	71%	29%
724	9,114	US 27 (Northwest) at Gilchrist County	6,562	2,552	72%	28%
725	6,327	US 441 (Northwest) at Columbia County Line	4,429	1,898	70%	30%

IE= Internal-External
EE= External-External
US = United States

2.2 External Validation Adjustments

In the absence of a travel survey, several iterations of adjustments were conducted to achieve a goal of 1.0 (volume-to-count ratio) at each external station. For external station 712, which is located on a minor collector road, due to the absence of any nearby count station, no count was coded. Traffic counts at adjacent links were used with appropriate adjustments for the external zone 710.

An Iterative Proportional Fitting process was used by creating a seed distribution and iteratively adjusting the cell values until a good match between the target row and column totals was achieved. For External-External trip tables, the row and column total targets represent the portions of the external station productions and attractions. For the External-Internal and Internal-External trips, productions and attractions are generated for both ends of the trip, the internal end of the trip, and the external end of the trip. The traffic counts acted as control total at each external zone such that the total number of vehicle trips (Internal-External, External-Internal, and External-External) generated at an external station is equal to the traffic count at the external station.

2.3 External Validation Results

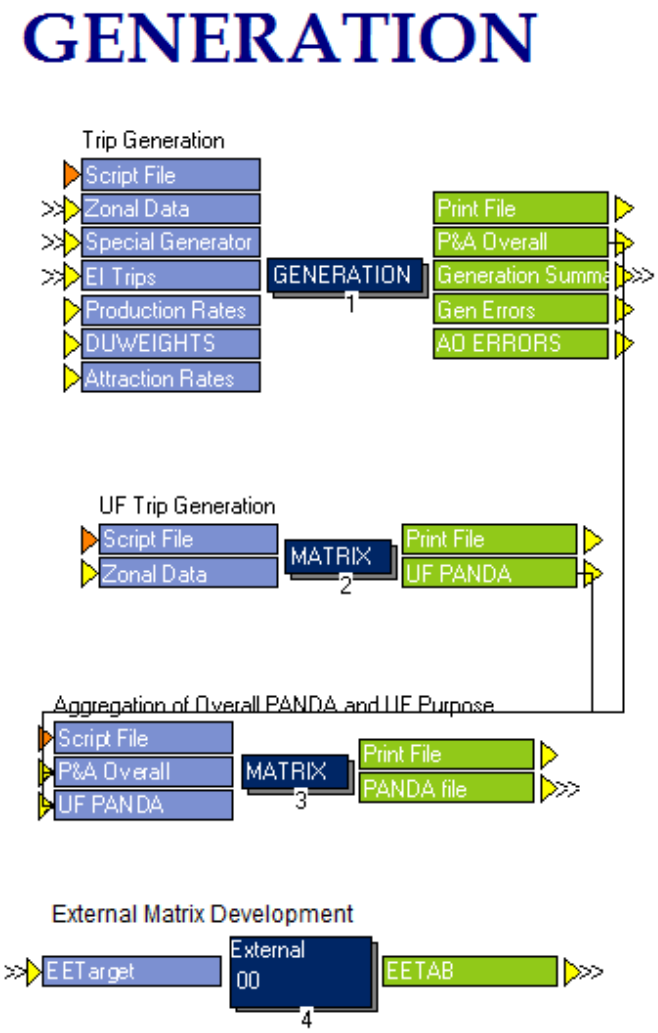
Final model validation results will be discussed in the highway assignment section. Assignment results indicate a reasonable match between the external travel movements estimated by the model and the traffic counts. The external cordon line achieves a volume-to-count ratio of 1.00. Corridors leading to or nearby external zones also were validated to satisfactory levels, based on this iterative adjustment process. The truck percent on I-75 and major arterials was resolved in the 2015 Gainesville model by providing truck-specific targets in the EETARGET file.

3. Trip Generation

3.1 Trip Generation Process

Trip Generation is the first step of the four-step travel demand forecasting process. In this step, the amount of travel by trip purposes is calculated. Trip generation focuses on the locations generating the travel, and not on the directionality of travel. Trip generation models provide the estimates of the number of trips by purpose produced by (trip production) or attracted to (trip attraction) a traffic analysis zone as a function of the demographic, socioeconomic, and land use characteristics of the zone. The most common forms of trip production and attraction models are cross-classification and linear regression. Cross-classification is generally used for trip production models and regression is used for trip attraction models. Figure 3.1 shows the trip generation model chain.

Figure 3.1 Trip Generation Model Chain



Trip Generation Process

The 2015 Gainesville Urbanized Area Transportation Study Model uses cross-classification trip production rates stratified by automobile availability (0, 1, 2, and 3+ automobile households), dwelling unit type (single-family, multifamily, and transient units), and household size (1, 2, 3, 4, and 5+ persons per household). Trip production rates for home-based work, home-based shop, home-based social/recreation, and home-based other purposes are shown in Table 3.1. The original source for these trip production rates was the North Florida Household Travel Survey, consistent with the previous 2010 Gainesville Urbanized Area Transportation Study Model. These trip production rates were not updated since there is no updated household travel survey available at the time of model development.

Trip attraction rates were originally derived from the 2005 Northeast Florida Regional Planning Model. As the Gainesville Urbanized Area Transportation Study Model and the Northeast Regional Planning Model are from Northeast Florida, both model regions share

similar socioeconomic characteristics, such as less of a reliance on tourism and seasonal residents than other parts of Florida. The trip attraction rates and dwelling unit weights are kept the same as the 2010 Gainesville Urbanized Area Transportation Study model, as shown in Table 3.2 and Table 3.3, respectively. Again, these were unchanged since an updated source of trip attraction rates is not available. The dwelling unit weights were retained from the 2010 Gainesville Urbanized Area Transportation Study Model.

The 2015 Gainesville Urbanized Area Transportation Study Model uses 11 trip purposes:

1. Home-based work;
2. Home-based shop;
3. Home-based social/recreation;
4. Home-based other (Home-based non-work, excluding university trips);
5. Non-home-based;
6. Home-based university;
7. University of Florida campus/dormitory;
8. Four-tire truck;
9. Single-unit truck;
10. Tractor-trailer; and
11. Internal-external.

Table 3.1 Trip Production Rates

Home-Based Work						
Dwelling Unit Type	Number of Automobiles Available	1	2	3	4	5+
Single Family	0	0.35	0.64	1.01	1.5	2.08
	1	0.69	0.98	1.35	1.84	2.42
	2	1.35	1.64	2.01	2.5	3.08
	3+	1.76	2.05	2.42	2.9	3.49
Multifamily	0	0.41	0.7	1.01	1.31	1.62
	1	0.95	1.49	2.02	2.56	3.1
	2	1.65	2.3	2.95	3.6	4.25
	3+	2.21	2.89	3.59	4.27	4.96
Hotel/Motel		1.04	0.72	0.5	0.39	0.39

Home-Based Shopping						
Dwelling Unit Type	Number of Automobiles Available	1	2	3	4	5+
Single Family	0	0.3	0.53	0.95	1.55	2.34
	1	0.59	1.02	1.55	2.18	2.89
	2	0.65	1.08	1.61	2.23	2.95
	3+	0.77	1.22	1.76	2.39	3.1
Multifamily	0	0.22	0.57	1.02	1.54	2.11
	1	0.5	0.95	1.4	1.83	2.27
	2	0.72	1.22	1.66	2.08	2.46
	3+	0.84	1.35	1.79	2.2	2.56
Hotel/Motel		0.33	1.43	2.2	2.75	3.19

Home-Based Social/Recreational						
Dwelling Unit Type	Number of Automobiles Available	1	2	3	4	5+
Single Family	0	0.21	0.28	1.28	1.47	2.2
	1	0.48	0.85	1.43	1.31	2.37
	2	0.53	0.89	1.85	2.07	2.77
	3+	0.7	1.07	2.04	2.24	2.97
Multifamily	0	0.18	0.63	1.08	1.53	1.98
	1	0.22	0.67	1.12	1.57	2.02
	2	0.64	1.09	1.54	1.99	2.44
	3+	0.84	1.29	1.74	2.19	2.64
Hotel/Motel		0.66	1.81	2.97	4.29	6.49

Home-Based Other						
Dwelling Unit Type	Number of Automobiles Available	1	2	3	4	5+
Single Family	0	0.29	0.64	1.67	3.38	5.78
	1	0.48	1.29	2.59	4.38	6.67
	2	0.62	1.79	3.34	5.2	7.33
	3+	0.68	1.94	3.58	5.59	7.99
Multifamily	0	0.35	0.78	2.28	4	6.23
	1	0.74	1.36	3.16	4.92	6.91
	2	1.12	1.87	3.71	5.59	7.34
	3+	1.17	2.09	4.05	5.75	7.56
Hotel/Motel		0.55	1.32	2.31	3.63	4.84

Table 3.2 Trip Attraction Rates

Purpose	Manufacturing	Other Industrial	Commercial	Service	Total	Dwelling Units	School Enrollment
Home-Based Work	0	0	0	0	1.8	0.5	0
Home-Based Shopping	0	0	6.1	0	0	0	0
Home-Based Social/ Recreational	0	0	0.5	0.5	0	1.61	0
Home-Based Other	0	0	1.5	1.5	0	0.3	1.5
Non Home-Based	0	0	3.54	1.71	0	0.3	0
Four-Tire Truck	0.47	0.55	0.45	0.22	0	0.13	0
Single-Unit Truck	0.12	0.15	0.13	0.04	0	0.05	0
Tractor-Trailer	0.05	0.09	0.04	0.01	0	0.02	0

Table 3.3 Dwelling Unit Weights

Average Persons Per Dwelling Unit	One-Person Households	Two-Person Households	Three-Person Households	Four-Person Households	Five-Person Households
0.00-1.12	0.89	0.11	0	0	0
1.13-1.37	0.76	0.22	0.02	0	0
1.38-1.62	0.59	0.34	0.05	0.01	0.01
1.63-1.87	0.46	0.34	0.11	0.06	0.03
1.88-2.12	0.32	0.36	0.16	0.11	0.05
2.13-2.37	0.24	0.36	0.18	0.14	0.08
2.38-2.62	0.21	0.33	0.19	0.16	0.12
2.63-2.87	0.12	0.35	0.19	0.23	0.11
2.88-3.12	0.13	0.34	0.18	0.16	0.19
3.13-3.37	0.12	0.29	0.18	0.17	0.24
3.38-3.62	0.08	0.24	0.2	0.2	0.28
3.63-3.87	0.05	0.2	0.19	0.23	0.33
3.88-4.12	0.04	0.16	0.17	0.24	0.39
4.13-4.37	0.02	0.15	0.14	0.21	0.48
4.38-4.62	0.01	0.15	0.13	0.17	0.54
4.63-5.99	0	0.05	0.07	0.14	0.74
6.00+	0	0	0.02	0.05	0.93

Home-based University and University of Florida Campus/Dormitory trip purposes are unique to the Gainesville Urbanized Area Transportation Study Model. These additional purposes also were used in the Gainesville Urbanized Area Transportation Study 2000, 2007 and 2010 models. It is important to capture these trips in a university town, such as the City of Gainesville. The Home-based University purpose is for trips traveling from off-campus housing to parking spaces within the University of Florida Campus. On the other hand, the University of Florida Campus/Dormitory trip purpose is trips from the University

of Florida on-campus dormitories to classrooms that are specified in the ZONEDATA file. It should be noted that the model has limited capabilities in simulating parking capacity beyond the number of parking spaces being stored in the ZONEDATA file and used in the attraction equations.

Home-based University and Dormitory trip production and attraction equations for the Home-based University and Dormitory trip purposes are listed below, as extracted from the model scripts. During validation, these trip rates were relocated to the Cube catalog keys (names depicted in {brackets}) to enhance model transparency. The text below shows the sample model script on how the catalog keys were used in this process. The green text with semicolons indicate the comments.

Home-Based University Productions:
RO.HBUP = {RATE_HBUP}*ZI.1.UF_OC_ST*; UF_OC_ST is off-campus (students); Default value of {RATE_HBUP} is 2.996*
Home-Based University Attractions:
RO.HBUA = {RATE_HBUA}*ZI.1.UF_PARKING*; PARKING is University of Florida Parking Spaces; Default value of {RATE_HBUA} is 1.375*
University of Florida Campus/Dormitory Productions:
RO.HDORMUP = {RATE_HDORMUP} *ZI.1.UF_DORM_ST*;UF_DORM_ST is Campus housing/Dormitory students; Default value of {RATE_HDORMUP} is 2.262*
University of Florida Campus/Dormitory Attractions:
RO.HDORMUA = {RATE_HDORMUA} *ZI.1.SEATS*; SEATS is University of Florida Classroom Seats ; Default value of {RATE_HDORMUA} is 0.7513*

3.2 Trip Generation Changes

The overall trip rates in the region seemed reasonable from the earlier version of the model. A few additional changes were made to better calibrate the University of Florida off Campus Trips generated.

Gainesville Urbanized Area Transportation Study Model has two Trip Generation Modules- Regular Trip Generation and the University of Florida Trip Generation. All student trips to the University of Florida are generated in the University of Florida Trip Generation. There are 24,202 off-campus students estimated to be living in the non-University of Florida Traffic Analysis Zones. These students make University of Florida trips at a rate of 2.2 trips per student. In addition, the earlier models considered the students in the general population. Thus, some of these trips have been overestimated in the earlier version of the model. To better calibrate student non-University of Florida trip generation, iterative calibration/validation runs were performed. To discount the overestimation of the non-University of Florida trips made by the students, the following adjustment was made. Instead of removing the entire student population from the general population (that would have been unrealistic since some students make non-University of Florida trips), about 40 percent of the student households were assumed to be making the non-University of Florida trips.

Another important change in the trip generation module was related to the University of Florida trip generation rates of the off-campus students. The 2010 Gainesville Urbanized

Area Transportation Study Model assumed a trip rate of 2.96 trips per off-campus student versus 2.2 trips per on-campus student. In coordination with the University of Florida staff, it was determined that the off-campus University of Florida trip rates are too high when compared to the on-campus University of Florida trip rates. Intuitively, more University of Florida trips per day are expected by the on-campus students due to their proximity to the campus. The off-campus students are expected to make their long trips and limit traveling back and forth between the University of Florida and their homes due to the longer travel times involved. Hence, the project team determined to reduce this trip rate during the calibration. A trip rate of 2.2 trips per student was assumed for the off-campus students as a result of the final calibration.

Additionally, changes were made in the trip generation module related to the student trip attractions. The off-campus students are attracted to the parking lots, instead of the classrooms. Hence, the location of student parking spaces becomes critical to the student trip destination. By extensive coordination with the University of Florida staff, the student parking lots along with the data were populated into the model. In the earlier versions of the model, both student and employee parking were combined and used in the trip generation. By separating student parking, students in the model distribution will now be only sent to the traffic analysis zones with student-parking.

3.3 Socioeconomic Data and Traffic Analysis Zone Structure

Socioeconomic data is a critical input to the trip generation process. As part of the 2015 effort, staff from the Metropolitan Transportation Planning Organization of the Gainesville Urbanized Area updated the 2010 socioeconomic data files to the new base year 2015, which were reviewed and refined by the consultant team in coordination with staff. Special generator and external trip files were updated by the consultant team.

Like the 2010 Gainesville Urbanized Area Transportation Study Model, the 2015 model study area covers all of Alachua County, including all nine municipalities within the county. The zonal structure of the 2010 model was reviewed by the consultant team and changes were performed as needed. The model has undergone extensive revisions in terms of the traffic analysis zone structure. The 2015 model has a total of 677 traffic analysis zones; 651 internal zones and 26 external zones within and outside study area. Figure 3.2 shows the Gainesville Urbanized Area Transportation Study travel demand model geographic coverage and the traffic analysis zone structure of the study area. Table 3.4 provides a comparison of the new 2015 traffic analysis zone structure with the old 2010 traffic analysis zone structure:

Table 3.4 2015 and 2010 Traffic Analysis Zone Structure

Year	Internal Zones	External Zones	Dummy Zones
2010	1-576	600 - 625	39, 91, 110, 111, 119, 129, 131, 145, 175, 212, 230, 333, 344, 353, 431, 443, 577-599
2015	1-651	700 - 725	652-699

Figure 3.2 2015 Traffic Analysis Zone Structure Map

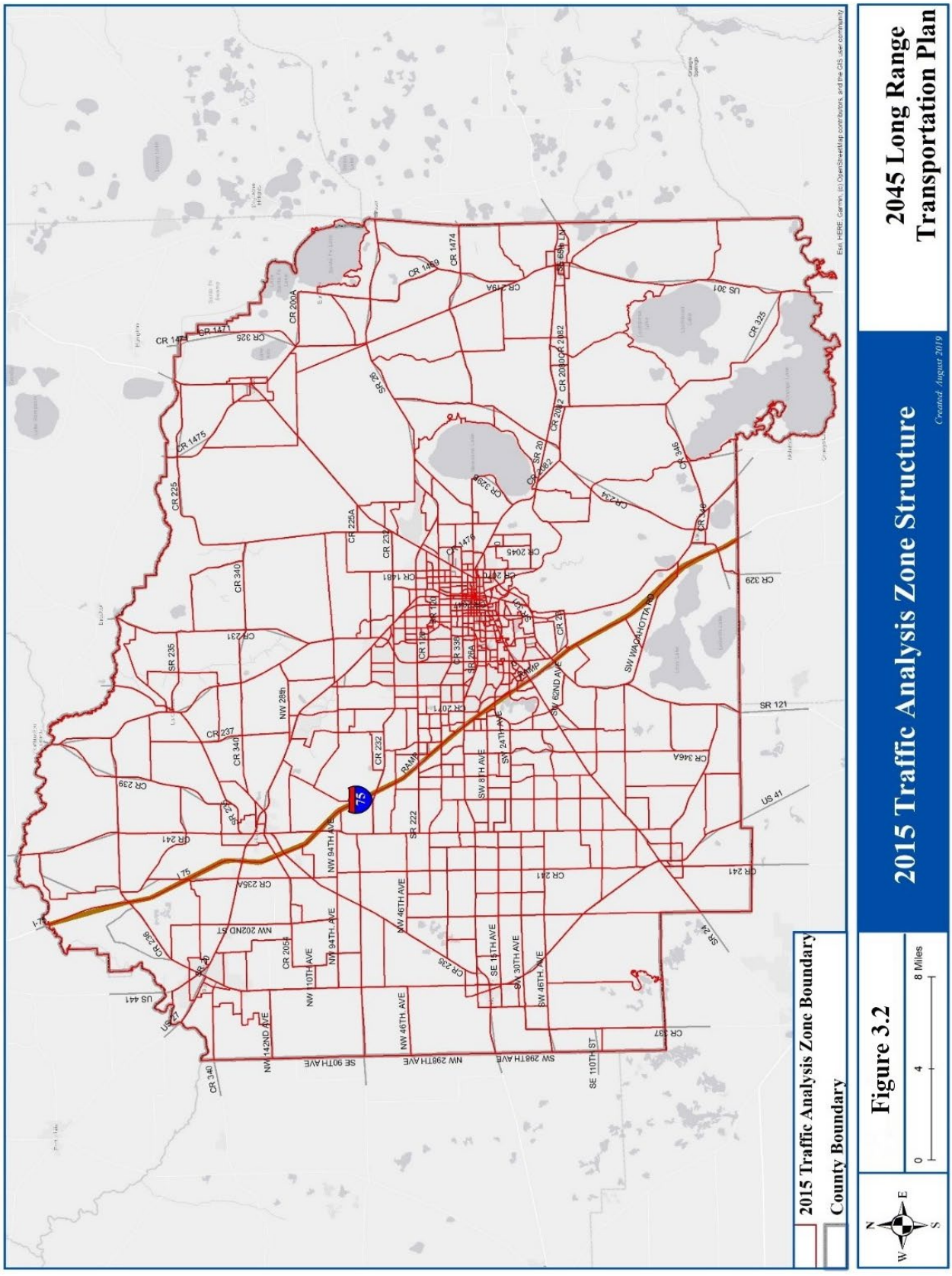
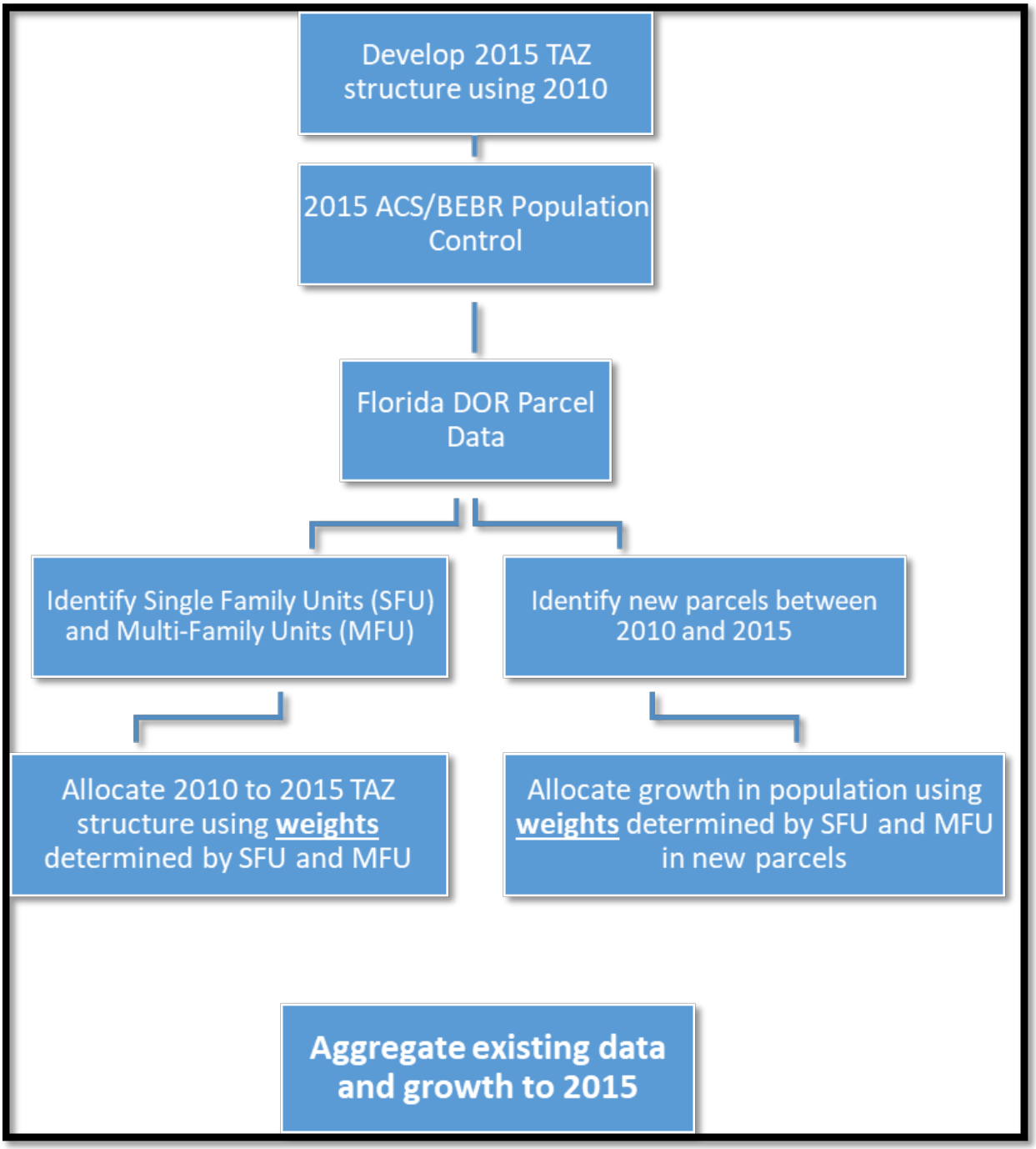


Figure 3.3 presents the correspondence between the 2010 traffic analysis zones and 2015 traffic analysis zones where the traffic analysis zones splits were made:

Figure 3.3 2015 Socioeconomic Data Development Flowchart



ACS = American Community Survey
BEBR = Bureau of Economic and Business Research
DOR = Florida Department of Revenue
TAZ = Traffic Analysis Zone

Figure 3.4 Traffic Analysis Zones Equivalency (2010 to 2015)

TAZ_2010	SPLIT_FLAG	TAZ2015
49	1	39
49	1	49
52	1	52
52	1	91
53	1	53
53	1	110
76	1	76
76	1	111
77	1	77
77	1	119
85	1	85
85	1	646
86	1	86
87	1	129
87	1	87
89	1	89
89	1	131
102	1	145
102	1	102
107	1	175
107	1	107
112	1	112
112	1	651
113	1	212
113	1	113
120	1	230
120	1	120
122	1	122
122	1	650
127	1	127
127	1	626
138	1	138
138	1	333
143	1	143
144	1	344
144	1	144
149	1	149
149	1	647

TAZ_2010	SPLIT_FLAG	TAZ2015
150	1	353
150	1	150
152	1	431
152	1	152
154	1	154
154	1	627
155	1	155
155	1	443
157	1	157
157	1	628
157	1	629
158	1	158
158	1	630
163	1	163
163	1	620
165	1	165
165	1	631
165	1	632
171	1	577
171	1	171
178	1	178
178	1	633
178	1	634
178	1	648
180	1	180
180	1	578
182	1	182
182	1	635
185	1	185
186	1	579
186	1	186
188	1	188
188	1	636
192	1	582
192	1	580
192	1	581
192	1	192
196	1	196
196	1	583
197	1	197
197	1	584

TAZ_2010	SPLIT_FLAG	TAZ2015
200	1	585
200	1	200
201	1	201
201	1	586
208	1	208
208	1	637
208	1	638
208	1	649
219	1	219
219	1	621
219	1	622
219	1	623
222	1	222
222	1	624
224	1	224
224	1	587
226	1	226
231	1	231
231	1	588
236	1	236
236	1	639
239	1	591
239	1	239
239	1	589
239	1	590
241	1	593
241	1	241
241	1	592
246	1	594
246	1	246
263	1	263
263	1	595
268	1	268
268	1	597
268	1	596
270	1	270
270	1	599
270	1	598

TAZ_2010	SPLIT_FLAG	TAZ2015
276	1	276
276	1	600
284	1	601
284	1	284
303	1	602
303	1	303
334	1	603
334	1	334
334	1	604
339	1	605
339	1	339
355	1	355
355	1	606
463	1	608
463	1	463
463	1	607
467	1	467
467	1	609
468	1	468
468	1	640
469	1	469
469	1	610
471	1	611
471	1	471
472	1	612
472	1	472
500	1	500
500	1	613
502	1	619
502	1	502
544	1	614
544	1	544
544	1	625
545	1	615
545	1	616
545	1	545
548	1	548
548	1	641
548	1	642

TAZ_2010	SPLIT_FLAG	TAZ2015
556	1	617
556	1	556
573	1	618
573	1	573
573	1	645
573	1	643
573	1	644

TAZ = traffic analysis zone

The consultant team in coordination with the Gainesville Urbanized Area Transportation Study team, developed the 2015 socioeconomic dataset which includes information on population disaggregated by single-family, multifamily, and hotel/motel units. It also provides information on automobile availability, property vacancy rates, and seasonal use. On the employment side, the dataset contains information disaggregated by service, commercial, manufacturing, and other industrial sectors. It also contains information on school enrollment, university employment, dormitory students, and parking. The parking costs in the 2015 dataset were adopted from the 2010 dataset based on discussions with staff from the Metropolitan Transportation Planning Organization for the Gainesville Urbanized Area.

2015 Traffic Analysis Zone Data Development.

The 2015 traffic analysis zone data was developed by reviewing the American Community Survey and the University of Florida, Bureau of Economic and Business Research control totals. Table 3.5 summarizes the employment, population and household growth controls within the region.

Table 3.5 Socio-economic Growth by Year

	Employment	Population	Households
2010	137,594	247,336	112,766
2015	154,640	253,316	115,697
Growth	17,046	5,980	2,931
Percent Growth	2.48	0.48	0.52

The socio-economic data was updated using the following procedure:

1. 2010 data was allocated to the 2015 traffic analysis zones using the traffic analysis zones correspondence.
2. Wherever the traffic analysis zones are split, the data was allocated using the traffic analysis zone split percentages (developed by reviewing single family and multifamily unit development density within each traffic analysis zone). The Florida Department of Revenue parcel data was used in this effort.
3. The growth has been allocated to the traffic analysis zones where new dwelling units between 2010 and 2015. The new dwelling units were used as "weights" to allocate the household growth.
4. The average household size was used to allocate the population growth.

The traffic analysis zone allocation method is illustrated in the following flowchart in Figure 3.3.

Figure 3.5 presents the revised traffic analysis zone structure. Figure 3.6 and Figure 3.7 show the thematic maps of the total population and employment of each traffic analysis zone within the travel demand model. The details of the socio-economic data format are included in Appendix A of this report and the data is summarized in Table 3.6.

Table 3.6 Year 2015 and 2010 Socio-economic Data Summary

Socio-Economic Data Summary	2015	2010
Permanent Population	253,316	247,336
Total Population	258,663	251,951
Permanently Occupied Dwelling Units	92,431	99,089
Transient and Permanently Occupied Dwelling Units	95,362	101,996
Total Service Employment	101,801	91,399
Total Commercial Employment	37,354	32,669
Total Manufacturing Employment	4,614	4,048
Total Other Industrial Employment	10,827	9,478
Total Employment	154,596	137,594
Permanent Population per Permanently Occupied Dwelling Unit	2.74	2.5
Total Population per Total Occupied Dwelling Unit	2.712	2.47
Total Employment per Permanent Population	0.61	0.556
Service to Total Employment	0.658	0.664
Commercial to Total Employment	0.242	0.237
Manufacturing to Total Employment	0.03	0.029
Other Industrial to Total Employment	0.07	0.069
Internal Person Trips per Permanently Occupied Dwelling Unit	11.84	11.63
Internal Person Trips per Total Occupied Dwelling Units	11.47	11.29
Internal Person Trips per Employee	7.078	8.372
Internal Person Trips per Person	4.319	4.657

Figure 3.5 2015 Traffic Analysis Zone Structure

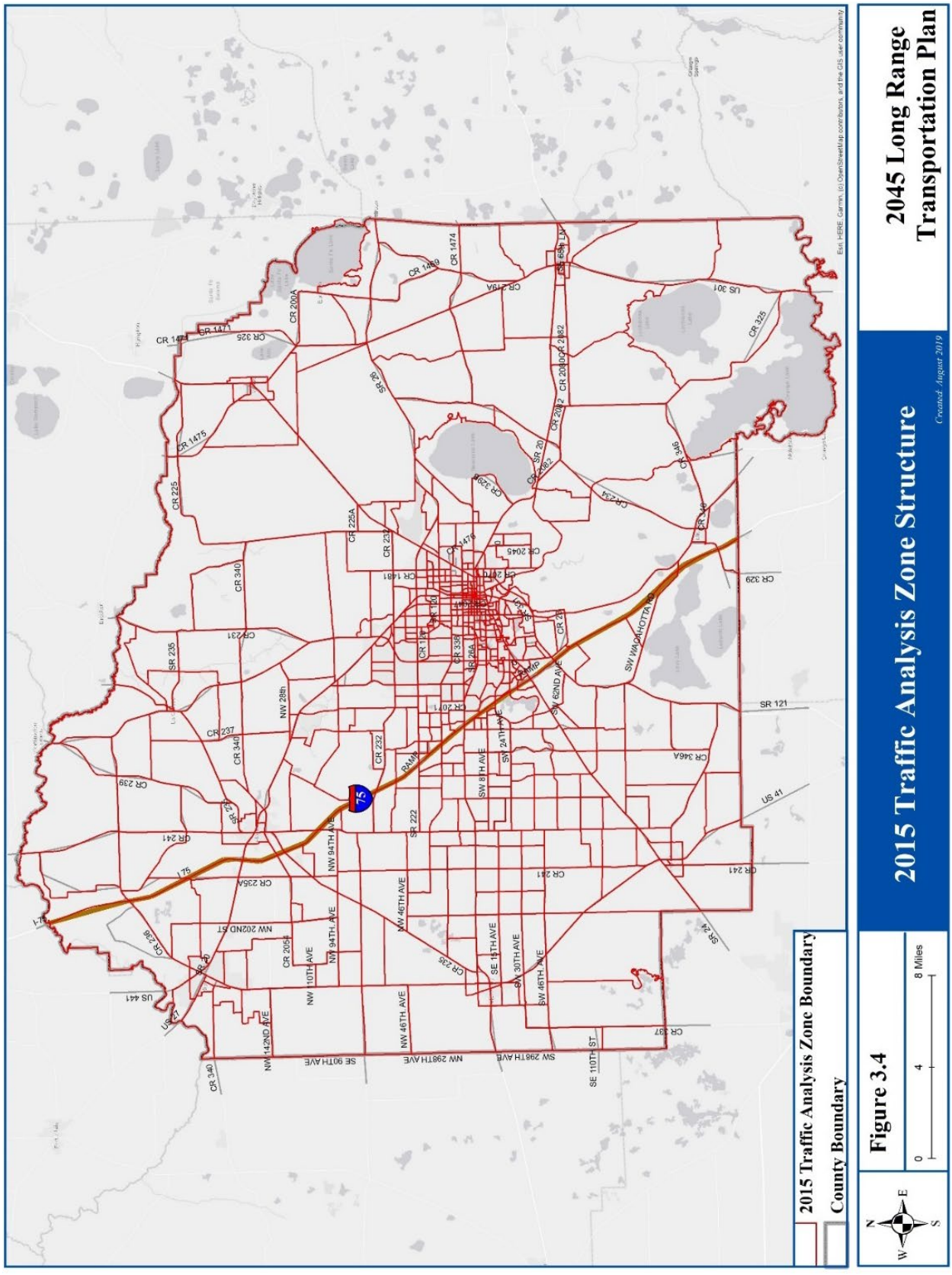


Figure 3.6 2015 Year Population by Traffic Analysis Zone

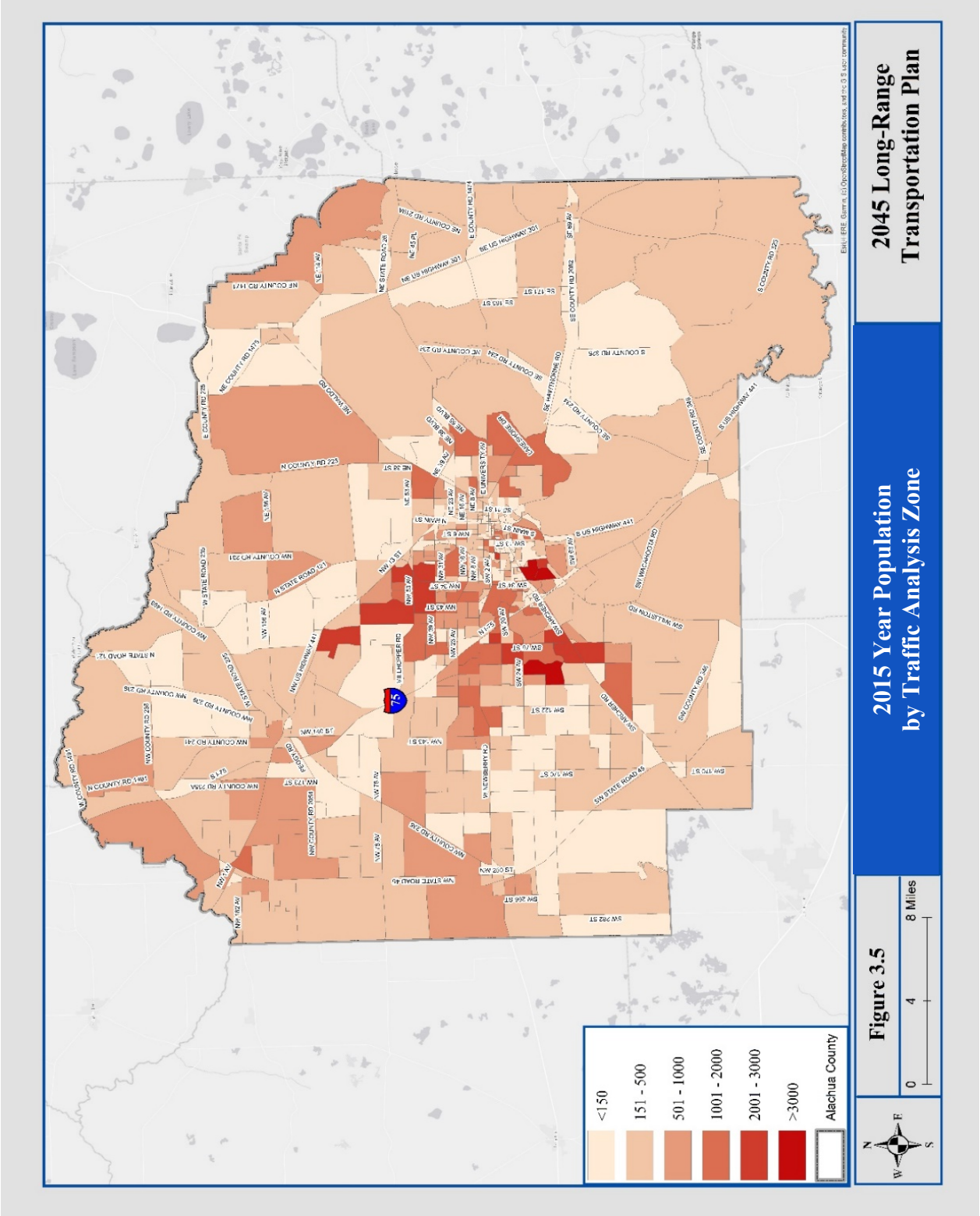
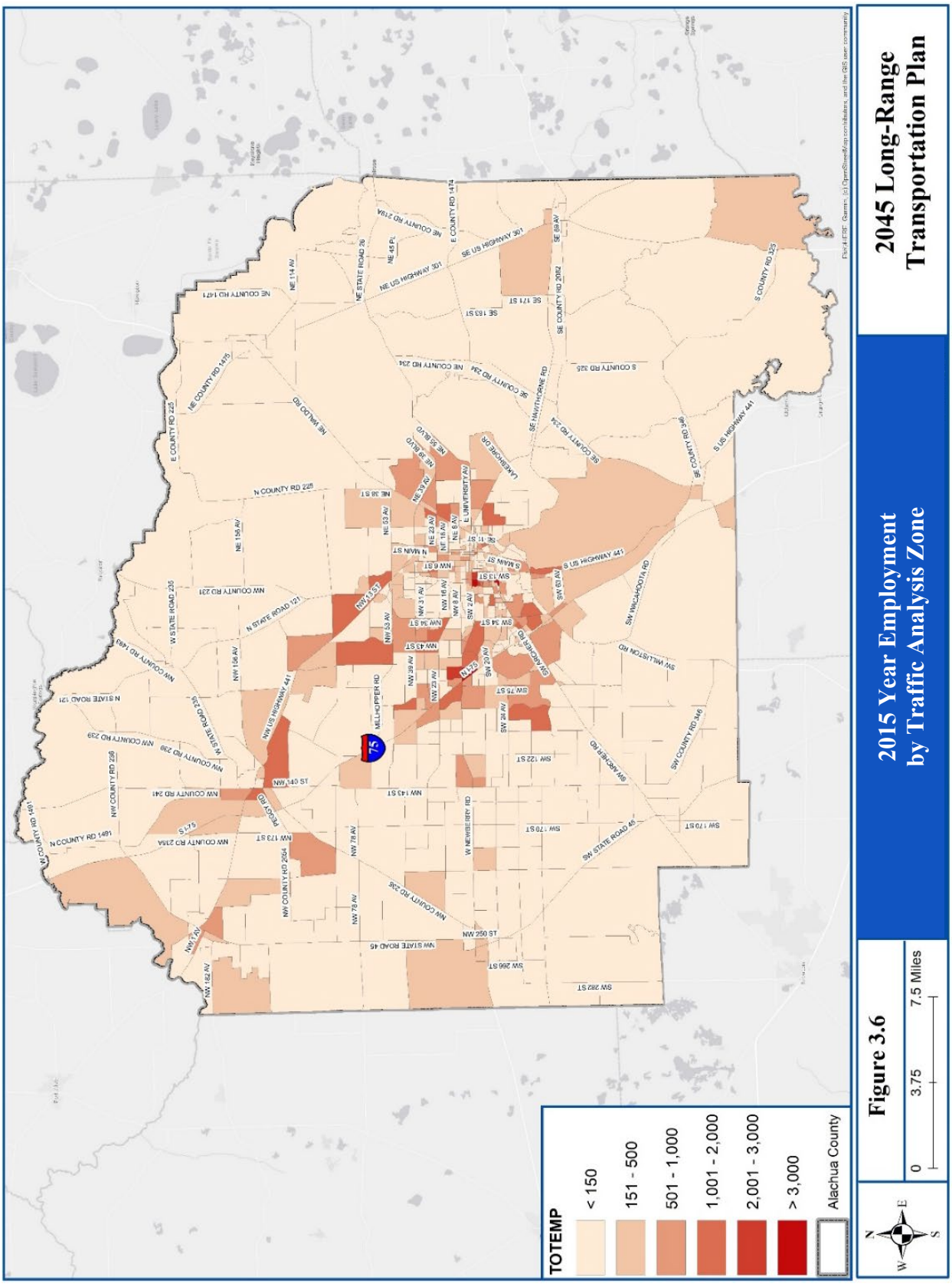


Figure 3.7 2015 Year Employment by Traffic Analysis Zone



3.4 Special Generators

There are certain traffic analysis zones where the standard trip generation algorithms alone cannot generate reasonable number of trips. These zones, because of the special land use characteristics, need special trip generation adjustments. These traffic analysis zones are called special generators. Best practice in travel demand forecasting is to minimize the use of special generators. Special generators should only be used where validation discrepancies exist and cannot be corrected with edits to other model files and parameters. The special generators are shown in Table 3.7.

Table 3.7 Special Generators

TAZ	PA dummy	Addition or Subtraction	Person Trips	HBW Percent	HBSH Percent	HBSR Percent	HBO Percent	NHB Percent	Description
536	A	+	27000	2%	2%	2%	92%	2%	Santa Fe College
440	A	+	655	20%	38%	38%	0%	4%	University of Florida
441	A	+	576	20%	38%	38%	0%	4%	University of Florida
433	A	+	408	20%	38%	38%	0%	4%	University of Florida
449	A	+	662	20%	38%	38%	0%	4%	University of Florida
453	A	+	1816	20%	38%	38%	0%	4%	University of Florida
460	A	+	362	20%	38%	38%	0%	4%	University of Florida

HBO = Home-based other
HBSH = Home-based shopping
HBSR = Home-based social recreation
HBW = Home-based work
NHB = Non home-based
PA = production/attraction
TAZ = traffic analysis zone

3.5 Trip Generation Validation Results

Throughout the validation process, trip generation statistics were summarized to assess model validity. Statistical comparisons were made between the 2010 Gainesville Urbanized Area Transportation Study Model and 2015 Gainesville Urbanized Area Transportation Study Model as well as model validation standards from the Florida Standard Urban Transportation Model Structure-Cube Framework Phase II: Model Calibration and Validation Standards Final Report. The trip generation validation statistics is summarized in Table 3.8 Percent Trips by Purpose and Table 3.9 Trip Generation Aggregate Rates. As shown by the results, the trip generation estimated by the model is within acceptable ranges.

Table 3.8 Percent Trips by Purpose

Purpose	Gainesville Urbanized Area Transportation Study 2010 Model		Gainesville Urbanized Area Transportation Study 2015 Model		FSUTMS*
	Production	Percent by Productions	Production	Percent by Productions	Percent by Productions
Home-Based Work	185,994	13.87%	170,371	13.26%	12-20%
Home-Based Shopping	135,822	10.13%	127,672	9.94%	10-20%
Home-Based Social-Recreation	120,838	9.01%	114,794	8.93%	9-12%
Home-Based Other	270,822	20.20%	254,071	19.77%	14-28%
Non home-Based	354,086	26.40%	343,226	26.71%	20-33%
Home-Based University	64,423	4.80%	72,745	5.66%	NA
Dormitory-Based University	22,767	1.70%	23,771	1.85%	NA
Truck-Taxi	84,379	6.29%	84,719	6.59%	NA
Internal-External	101,877	7.60%	93,480	7.28%	NA
Total	1,341,008	100.00%	1,284,849	100.00%	NA

** Table F.1 – Florida Standard Urban Transportation Model Structure Cube Framework Phase II Model Calibration and Validation Standards*

Table 3.9 Trip Generation Aggregate Rates

Trip Generation	Gainesville Urbanized Area Transportation Study 2015 Model	Gainesville Urbanized Area Transportation Study 2010 Model	FSUTMS*
Persons/Dwelling Units	2.71	2.47	2.00 - 2.70
Employment/Person	0.61	0.56	0.35 - 0.75
Internal Person Trips/Dwelling Units	11.9	11.63	NA
Internal Person Trips/Person	4.341	4.66	NA
Internal Person Trips/Employment	7.111	8.37	NA

** Table F.1 - Florida Standard Urban Transportation Model Structure Cube Framework Phase II Model Calibration and Validation Standards*

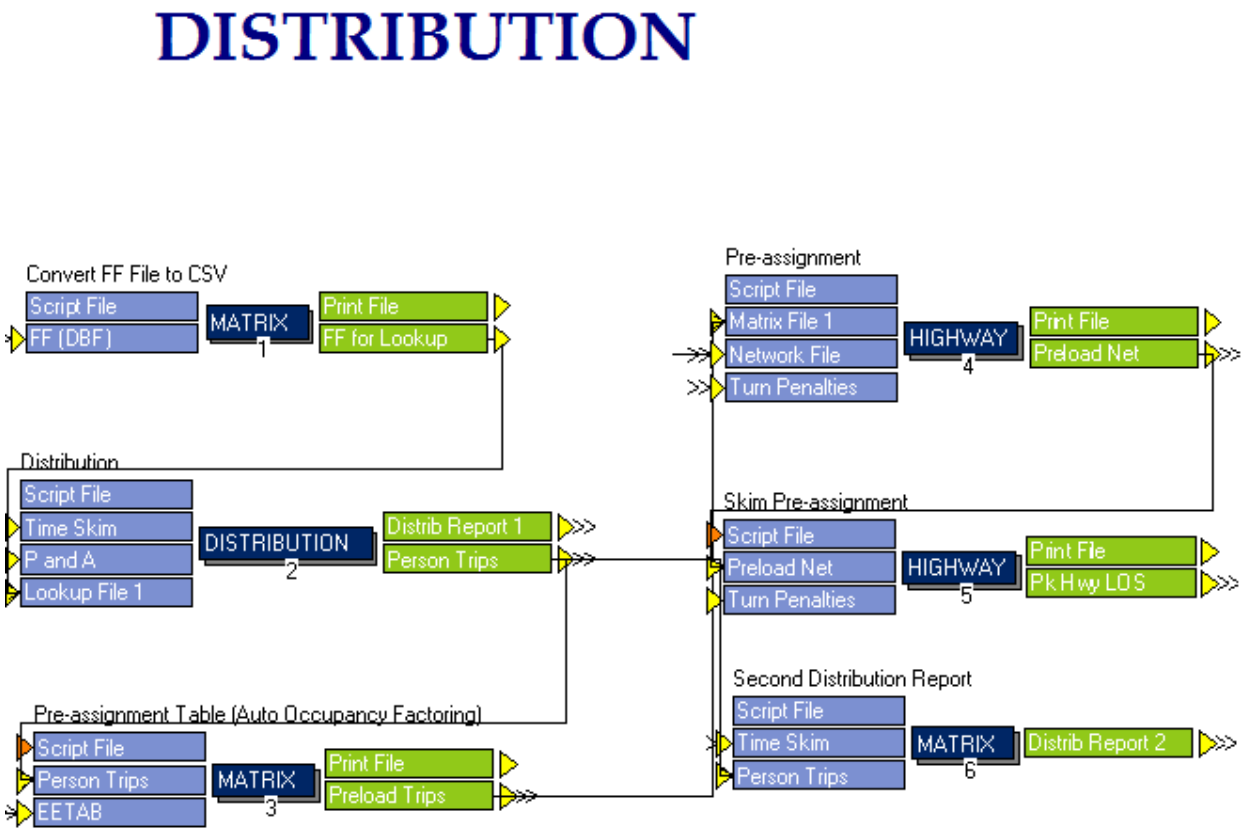
4. Trip Distribution

Trip distribution models link trip productions and attractions between pairs of traffic analysis zones. The most common method for trip distribution in four-step models is the gravity model. The trip distribution step distributes all trips produced in a zone to all possible attraction zones using the gravity model. The gravity model uses the trip productions and attractions estimated in the trip generation model and the network friction to distribute the trips.

Friction factors are used to represent the effects of travel impedance. Friction factors can be computed by a formula, such as a gamma function, or can be calibrated such that a modeled trip length frequency distribution matches an observed trip length frequency distribution.

Trip distribution results in person-trip matrices detailing the trips between each production and attraction traffic analysis zone. These person-trips matrices are later processed in the model chain during mode choice to allocate trips by automobile occupancy and transit categories. Figure 4.1 shows the trip distribution model chain.

Figure 4.1 Trip Distribution Model Chain



For evaluation of trip distribution, the *Federal Highway Administration Travel Model Validation and Reasonability Checking Manual* recommends two general types of aggregate checks of trip distribution model results - trip length checks and origin-destination pattern checks. For this model, the evaluation was accomplished by comparing statistics for average trip length and the percentage of intrazonal trips between the 2015 Gainesville Urbanized Area Transportation Study Model, the 2010 Gainesville Urbanized Area Transportation Study Model, and the Florida Standard Urban Transportation Model Structure standards. Additionally, desire line maps were prepared to show travel movements between zones, and trip length-frequency distributions were reviewed for reasonableness.

4.1 Trip Distribution Model Structure

The general distribution process includes the building of highway networks and travel time skims as well as the application of the gravity model. The elements of these processes are described below.

Building Highway Network

The Florida Standard Urban Transportation Model Structure includes a module known as "Highway Network" to process highway networks of model areas. As part of the model validation process, the consultant conducted an in-depth review of the highway network. Network characteristics (number of lanes, area type, and facility type) were updated to reflect 2015 conditions of the roadway system throughout Alachua County.

Travel Time Skims

Free-flow travel time skims between zone pairs are developed as the last step in the "Highway Network" module in the Florida Standard Urban Transportation Model Structure, including the updating of travel time skims with intrazonal and terminal times. Highway network characteristics are an input to this process. In addition to the highway network characteristics, other input files are generally used during network skimming.

The first of these is the turn penalties file which contains a record of all prohibited turning movements in the network. Turning movements were reviewed to include any updated prohibited movements for the year 2015 conditions during validation. The turn penalty file can also include time penalties; however, time penalties were not recommended in the model area as the highway assignment validated reasonably well without supplemental travel time factors.

Also, a toll file is used in most Florida models to identify toll plaza characteristics. However, because no toll roads exist in Alachua County, this file is not used in the 2015 Gainesville Urbanized Area Transportation Study Model.

Intrazonal times represent the travel time it takes to travel within or across a zone. These times are calculated as one-half of the average travel time from each zone to the two nearest adjacent zones. Terminal times represent the time required at either end of a trip to travel from an origin to a vehicle or from the vehicle to the destination. More specifically, this accounts for the time necessary to walk to or from the vehicle used for any given trip. Terminal times are typically highest in central business districts and lowest in residential areas. Table 4.1 lists the terminal times by area type used in the 2015 Gainesville Urbanized Area Transportation Study Model.

Table 4.1 Terminal Times

Terminal Times*	Area Type	Area Type Descriptions
5	12	Urbanized Area (under 500,000) Primary City Central Business District
5	13	Other Urbanized Area Central Business District and Small City Downtown
5	14	Non-Urbanized Area Small City Downtown
3	21	Central Business District Fringe Areas
3	22	Industrial
1	31	Residential Area of Urbanized Areas
1	32	Undeveloped Portions of Urbanized Areas
1	33	Transitioning Areas/Urban Areas over 5,000 Population
2	42	Other Outlying Business District
1	51	Developed Rural Areas/Small Cities under 5,000 Population
1	52	Undeveloped Rural Areas

**Terminal Times listed in whole minutes*

Trip Distribution Module

The "DISTRIBUTION" module distributes trips between zones using a gravity model. The primary input data used for DISTRIBUTION are the friction factor file and a set of congested highway skims. These are used by the gravity model to measure the effects of spatial separation between zones for trip distribution. It is generally assumed that productions are less likely to be linked to destinations with greater travel times if alternative destinations with lesser travel times and similar attractiveness are available.

Friction factors from the 2010 Gainesville Urbanized Area Transportation Study Model were used for the 2015 Gainesville Urbanized Area Transportation Study Model without modification. Since no new household travel surveys were conducted since the 2010 model calibration and validation, and the resulting trip distribution was reasonable, it was decided to maintain the existing friction factor set in the 2015 model.

Friction factors are used by the gravity model to link the trip productions and attractions generated by GENERATION. These trip interchanges denote person trips traveling from one zone in the model to another. Trips are distributed by the trip purposes found in the 2015 Gainesville Urbanized Area Transportation Study Model. These person trips are later converted into vehicle trips during mode choice and then loaded onto networks during highway and transit assignment. The next subsection describes the checks, modifications and adjustments made to trip distribution assumptions to verify and improve model validity.

4.2 Trip Distribution Model Development and Validation

Errors in the trip distribution phase can lead to significant problems in the execution of the following steps in the model chain (i.e., mode choice and trip assignment). Hence, efforts were taken to maximize the accuracy of the 2015 Gainesville Urbanized Area Transportation Study Model trip distribution module. These efforts included adjustments to network speeds and capacities and corrections of network link attributes.

Speeds and Capacities

The 2015 speed and capacity lookup table are the same as the 2010 model; the main effort for validating the capacity was to update the number of lanes for the highway network. The Florida Department of Transportation Roadway Characteristics Inventory has been used as the primary source for updating the number of lanes to represent the 2015 conditions.

Penalties and Prohibitions

The TURN.pen file, formerly known as TCARDS, allows for the adjustment of travel times on specific links by either including a time penalty to pass from one link to another or by prohibiting the movement altogether. Prohibitions are confined to ramps located along Interstate 75, mainly to guide trips to the correct ramps for each travel movement. No time penalties were added during the 2015 model validation effort. The TURN.pen file is listed in Appendix B.

Friction Factors

Friction factors are used in the gravity model to represent the effects of travel impedance. These factors define the measure of separation based on travel impedances between traffic analysis zones. The friction factors from the 2010 model were reviewed and no updates were necessary for the 2015 model validation. Analysis of the trip distribution summaries did not indicate a need for modification to improve trip distribution. Average trip lengths seemed reasonable, intrazonal percentages made sense, and aggregate trip distribution patterns looked logical. Furthermore, there are no updated household travel diary survey data for Alachua County to allow for the calibration of new friction factors. A copy of the validated friction factor file (FF.dbf) is listed in Appendix C.

4.3 Trip Distribution Validation Results

Based on the Florida Standard Urban Transportation Model Structure standards, three gravity model checks were developed. They are average trip length by purpose, percentage of intrazonal trips, and trip flow patterns/ trip length frequency distribution.

Average Trip Length by Purpose

Table 4.2 shows a comparison of average trip length statistics generated by the 2010 and 2015 Gainesville Urbanized Area Transportation Study models and the Florida Standard Urban Transportation Model Structure standards. Comparisons between the 2015 and 2010 Gainesville Urbanized Area Transportation Study models show a small trip length increase in home-based university, truck-taxi, and Internal-External. Overall, the trip lengths are very similar to the 2010 model. Although the average trip lengths are on the low side compared to the Florida Standard Urban Transportation Model Structure standards, this is to be expected given the relatively smaller region covered by the Gainesville Urbanized Area Transportation Study Model.

Table 4.2 Average Trip Lengths (in Minutes)

Trip Purpose	Alachua County		FSUTMS*
	2015	2010	
Home-Based Work	15.09	14.67	15-28
Home-Based Shop	13.41	13.09	10-18
Home-Based Social/Recreation	12.77	12.49	11-19
Home-Based Other	13.45	13.24	10-20
Non-home-Based	10.86	10.51	10-18
Home-Based University	9.19	9.31	NA
University of Florida Campus/Dorm	6.21	6.2	NA
Truck-Taxi	15.74	15.4	12-20
Internal-External	26.27	25.77	27-45

** Table F.1 - Florida Standard Urban Transportation Model Structure Cube Framework Phase II Model Calibration and Validation Standards*

Intrazonal Trip Distribution

Comparisons between the 2010 and 2015 Gainesville Urbanized Area Transportation Study models indicate that the percentage of intrazonal trips is similar for most trip purposes. These results are illustrated in Table 4.3.

Table 4.3 Intrazonal Trip Summary

Purpose	Alachua County 2015			Alachua County 2010			FSUTMS *
	Total Trips	Intrazonal Trips	Percent	Total Trips	Intrazonal Trips	Percent	
Home-Based Work	170,371	2,354	1.38%	185,977	2,640	1.42%	1-4%
Home-Based Shop	127,672	4,020	3.15%	135,794	4,894	3.60%	3-9%
Home-Based Social/Recreation	114,794	7,828	6.82%	120,814	8,953	7.41%	4-10%
Home-Based Other	254,071	10,105	3.98%	270,787	10,457	3.86%	3-7%
Non-home-Based	343,225	23,121	6.74%	354,049	24,913	7.04%	5-9%
Home-Based University	72,745	48	0.07%	64,423	35	0.05%	NA
University of Florida Campus/ Dormitory	23,771	657	2.76%	22,767	1,336	5.87%	NA
Truck-Taxi	84,718	1,135	1.34%	84,055	1,290	1.53%	NA
Internal-External	93,479	0	0.00%	101,867	0	0.00%	NA
Total	1,284,844	49,268	3.83%	1,340,533	54,518	4.07%	3-5%

** Table F.1 - Florida Standard Urban Transportation Model Structure Cube Framework Phase II Model Calibration and Validation Standards*

Trip Flow Patterns and Trip Length Frequency Distributions

The model area trip flow pattern analysis is another aggregate visualization check in the validation process. traffic analysis zone to traffic analysis zone trip flows were conducted and checked with local knowledge to verify the reasonableness of trip distribution. In Figure 4.2, traffic analysis zone to traffic analysis zone daily total trip flows from the model are represented by desire lines. Only daily trip flows that are greater than 100 trips are shown. The concentration of trip desires in the downtown area and general radial patterns makes intuitive sense given the land-use patterns of Alachua County. Figures 4.3 through 4.6 show the trip length distribution by purpose - the 2010 and 2015 trip length-frequency distributions are found to be very similar.

In summary, the 2015 trip distribution validation results are consistent with the 2010 model and within acceptable Florida Standard Urban Transportation Model Structure standards.

Figure 4.2 Desire Lines by Traffic Analysis Zone

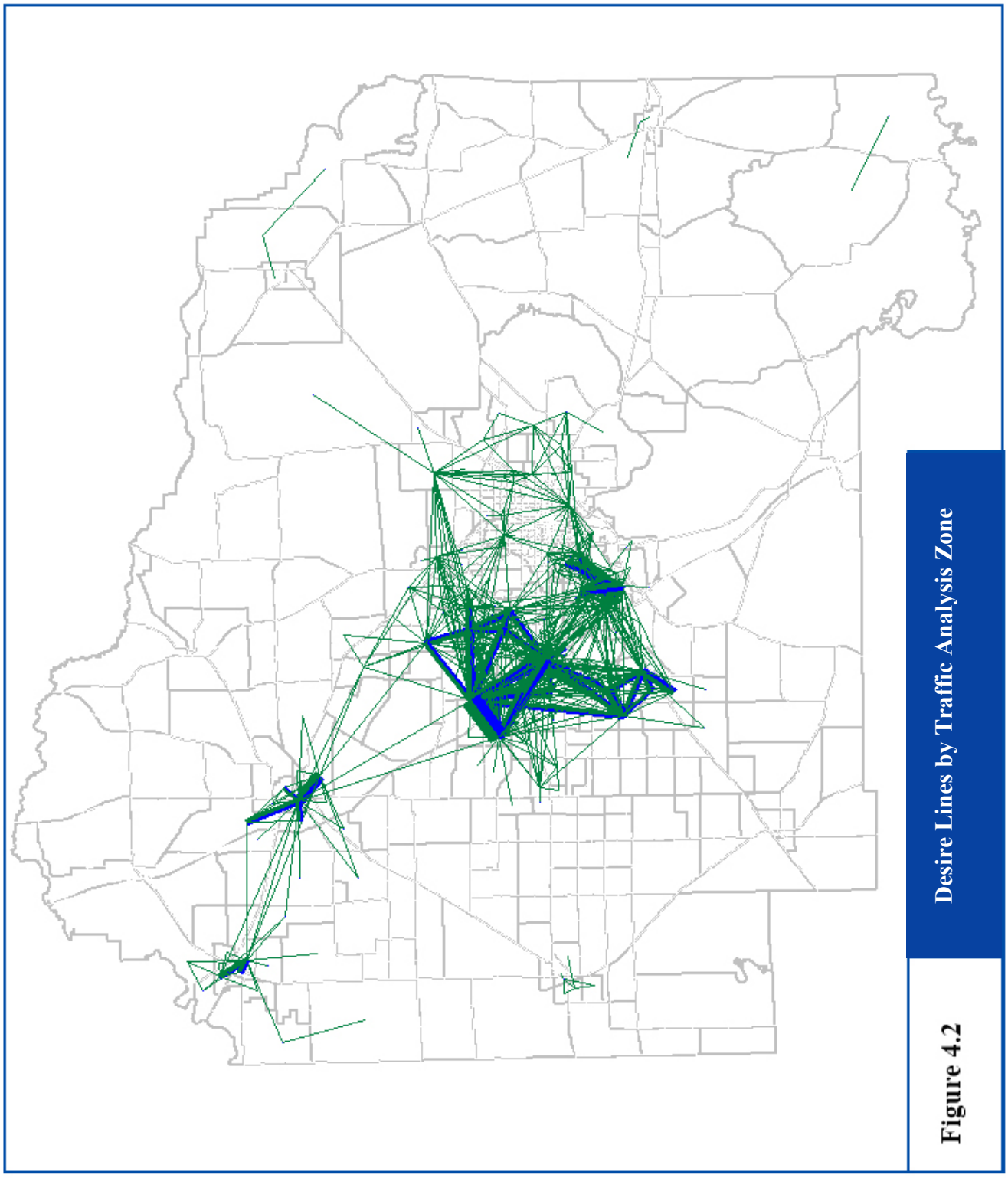


Figure 4.3 Trip Length Frequency Distribution for Home-Based Work

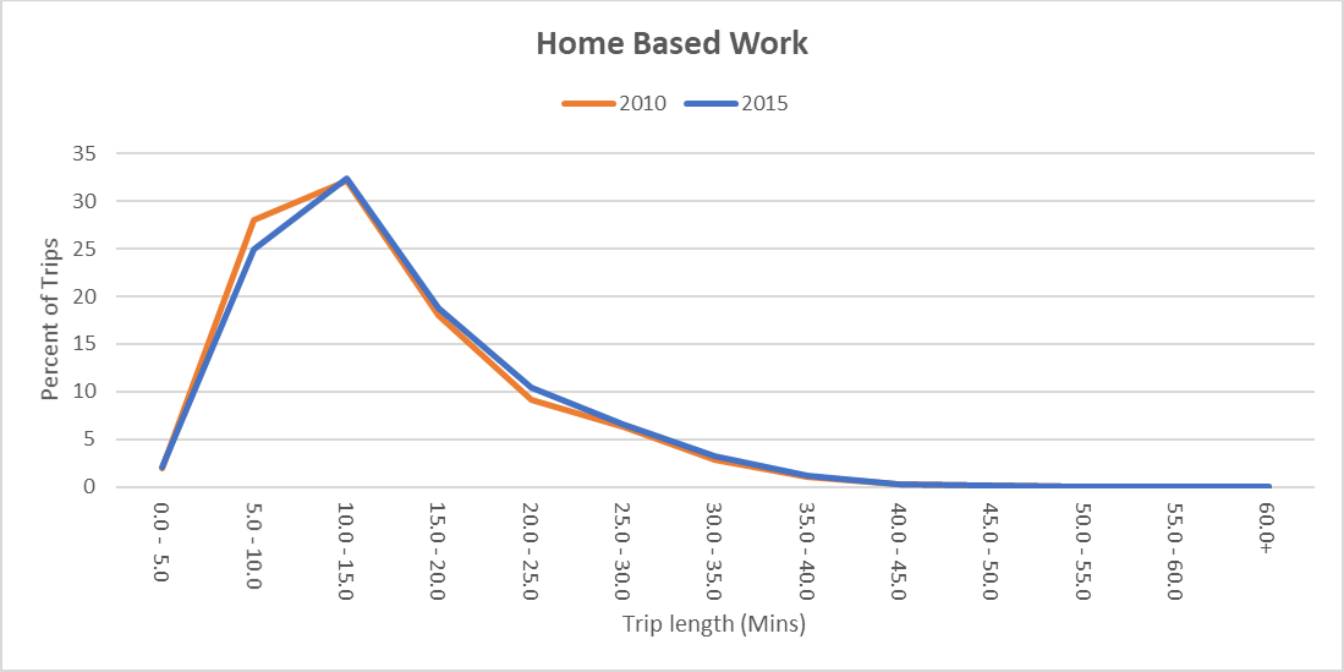


Figure 4.4 Trip Length Frequency Distribution for Home-Based Shopping

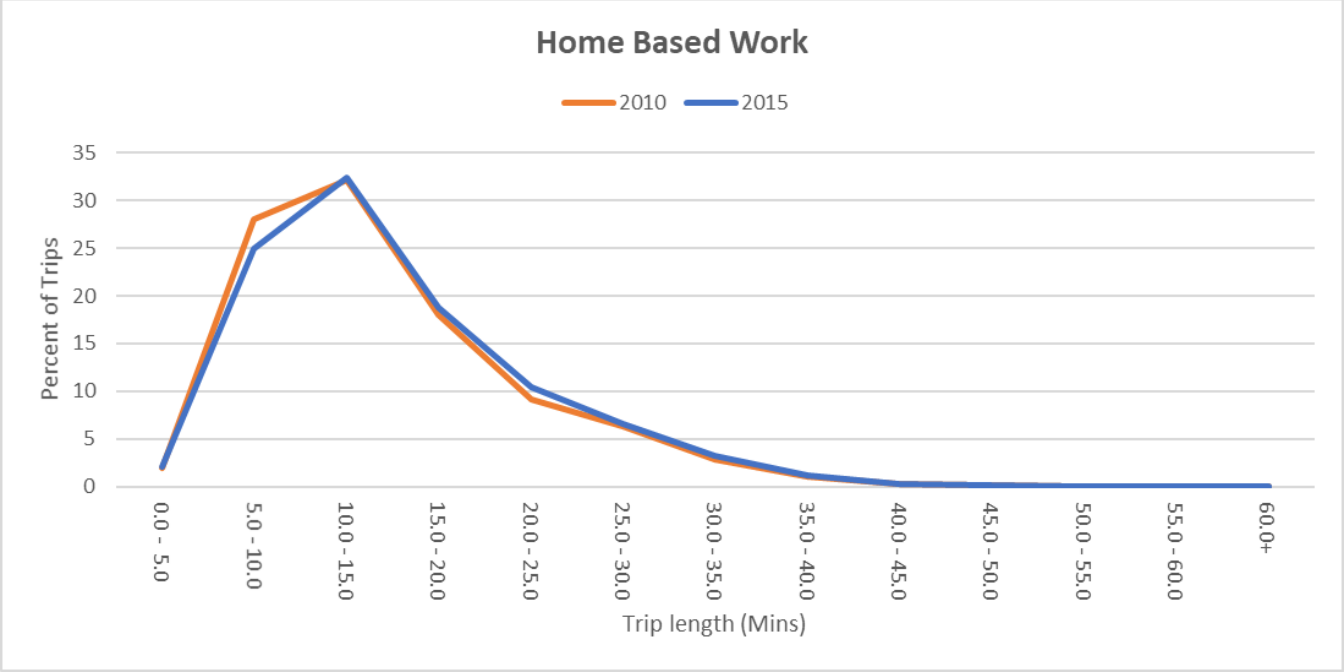


Figure 4.5 Trip Length Frequency Distribution for Home-Based Social/Recreational

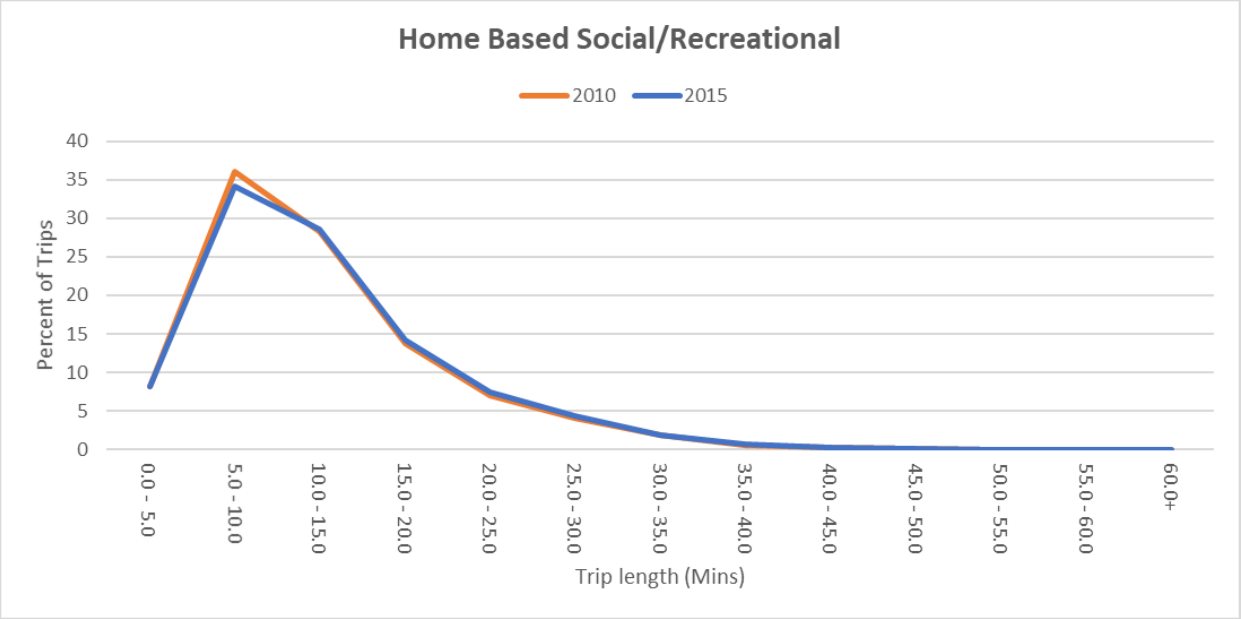
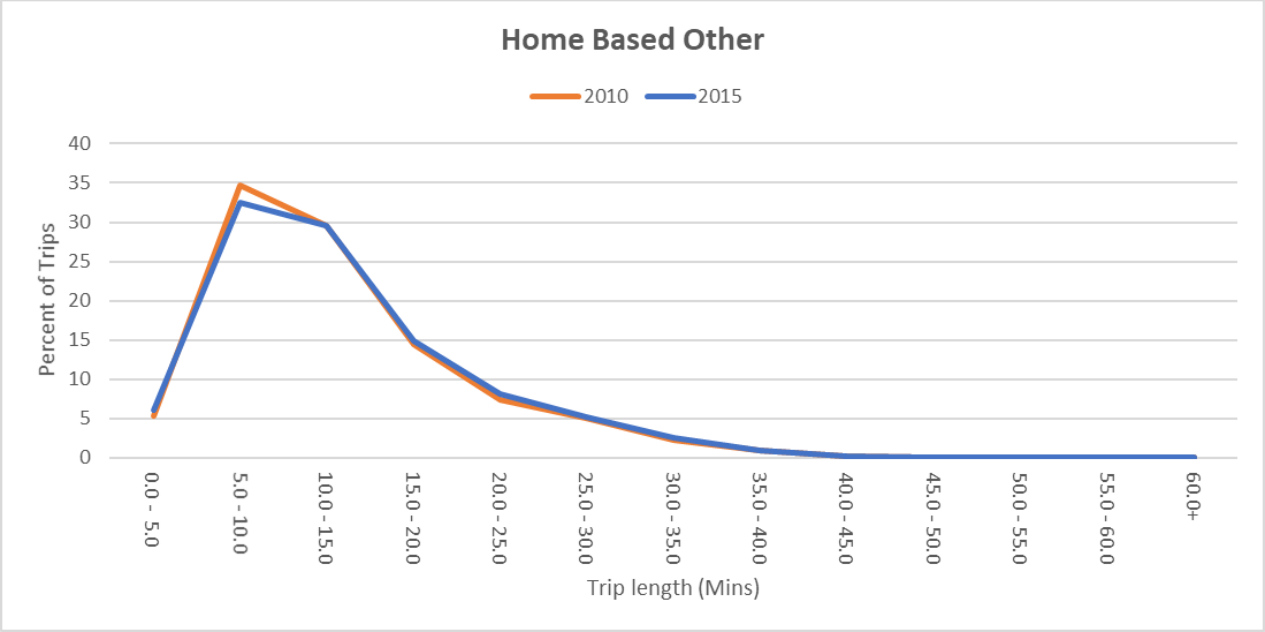


Figure 4.6 Trip Length Frequency Distribution for Home-Based Other

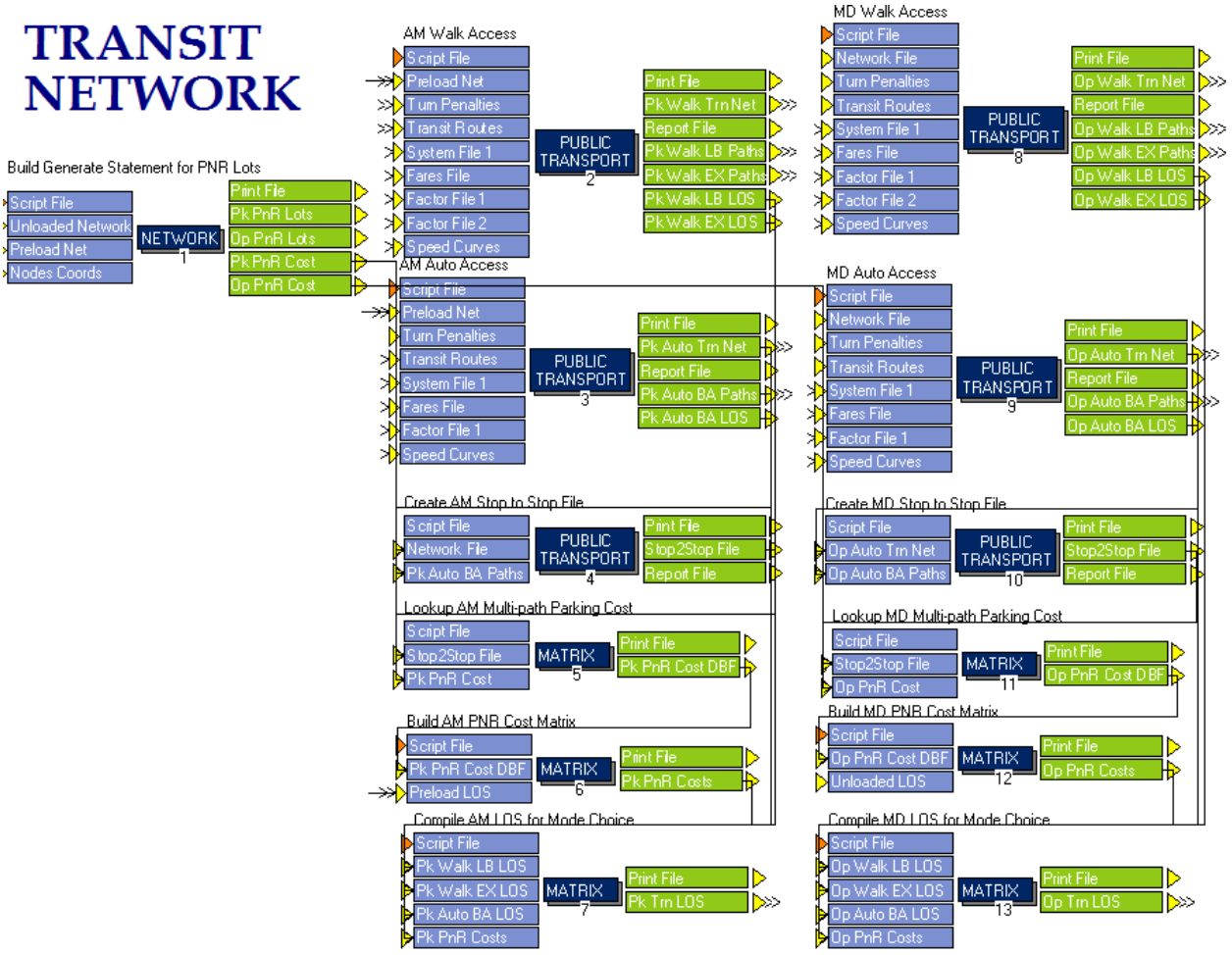


5. Transit Accessibility and Path-Building

Integral to the building of a transit network is the availability of access to transit. A critical component of transit access is identifying the zones that are within an acceptable walking distance to a transit stop. Walk access is generally provided from centroids to stops within a specified distance. Transit path-building involves transit fares, transit routes, and stops/stations.

These inputs are used for each of the transit modes during peak and off-peak periods used in the model. The 2015 Gainesville Urbanized Area Transportation Study Model currently only has one transit mode, local bus, but has the capability of expansion to other transit modes in the future. Figure 5.1 shows the structure of the transit network module.

Figure 5.1 Transit Network Module



5.1 Transit Access and Path-Building Module Structure

The Florida Standard Urban Transportation Model Structure transit network module is used to establish transit access and path-building. The module constructs separate peak-period(AM) and off-peak(MD) transit networks using congested highway networks as an

input to produce transit vehicle travel times. Transit path-building involves the generation of transit path skim matrices, fares, and station-to-station interchanges.

Transit accessibility is represented by each zone's pedestrian environment variables that are stored in the ZONEDATA file. The pedestrian environment variables define several factors that are essential to have sufficient accessibility to bus stops, such as sidewalk availability, ease of street crossing, non-motorized connections, and building setbacks. Each variable is given a score between 0 and 3 and the accumulated scores of the four pedestrian environment variables are saved as "SUM" in the ZONEDATA file, which ranges from 0 to 12. Future changes to the zonal transit accessibility will require modification of pedestrian environment variable scores as well as updating "SUM" values to get total pedestrian environment variable scores for each traffic analysis zone. Table 5.1 indicates what each pedestrian enhancement variable value represents. These variables and categories remain unchanged from the 2010 model.

Table 5.1 Pedestrian Environment Variables

Variables	PEV = 0	PEV = 1	PEV = 2	PEV = 3
Sidewalk Availability	No sidewalks	<10 percent have sidewalks	10 to 90 percent have sidewalks	>90 percent have sidewalks
Ease of Street Crossing	Crossing difficult	<10 percent have easy crossing	10 to 90 percent with easy crossing	>90 percent with easy crossing
Non-motorized Connections	No connections	<10 percent have connections	10 to 90 percent have connections	>90 percent have connections
Building Setbacks	All large setbacks	<10 percent have minimum setbacks	10 to 90 percent have minimum setbacks	>90 percent have minimum setbacks

5.2 Transit Access and Path-Building Model Development and Validation

Most of the effort in validating the transit accessibility and path-building focused on ensuring that the transit network was up to date and accurately reflected base year conditions. Also, walk access links were checked to ensure adequate connectivity.

Transit Accessibility

The key effort in validating the transit network consisted of three parts. The first was to review the existing transit network and update it to the year 2015 operating conditions. This required significant additions and rerouting of the transit lines. The second part was to check that there is no "Later Gator" bus route coded around the University of Florida campus since this may result in over assignment. Finally, pedestrian environment variables in the ZONEDATA file were reviewed and updated for each zone. However, because of limited data availability, the pedestrian environment variables are generally kept the same as in 2010.

Transit Fare

The City of Gainesville Regional Transit System transit fare was $1.50 during the year 2010, used for the Gainesville Urbanized Area Transportation Study base year 2010 model validation. In 2015, there was no fare change, according to the Gainesville Regional Transit System. The fare has been set to $1.50 for the base year 2015 in ALACHUA.FAR file. For the future year transit analysis, the bus fare factor (BUSSFAREFAC) value may need to be updated to represent fare increases beyond inflation.

Transit Routes

As validation efforts moved towards reasonable transit assignments, transit network rerouting, bus stop locations and local zonal access to bus stops were reviewed using the Gainesville Regional Transit System General Transit Feed Specification files. Headway data were also extracted from the Gainesville Regional Transit System General Transit Feed Specification files and the transit route file (troute15.lin) was updated.

The transit route file had to be overlaid on the highway network when new bus locations were added due to the necessity of splitting highway links where a bus stop exists. The highway network was therefore updated at the same time as the transit route file was updated. "Later Gator" bus routes are not included in the 2015 Gainesville Urbanized Area Transportation Study Model as these are evening bus services specifically for the University of Florida students and operate for only limited hours while the model is designed to estimate daily peak and off-peak transit ridership.

5.3 Transit Access and Path-Building Model Validation Results

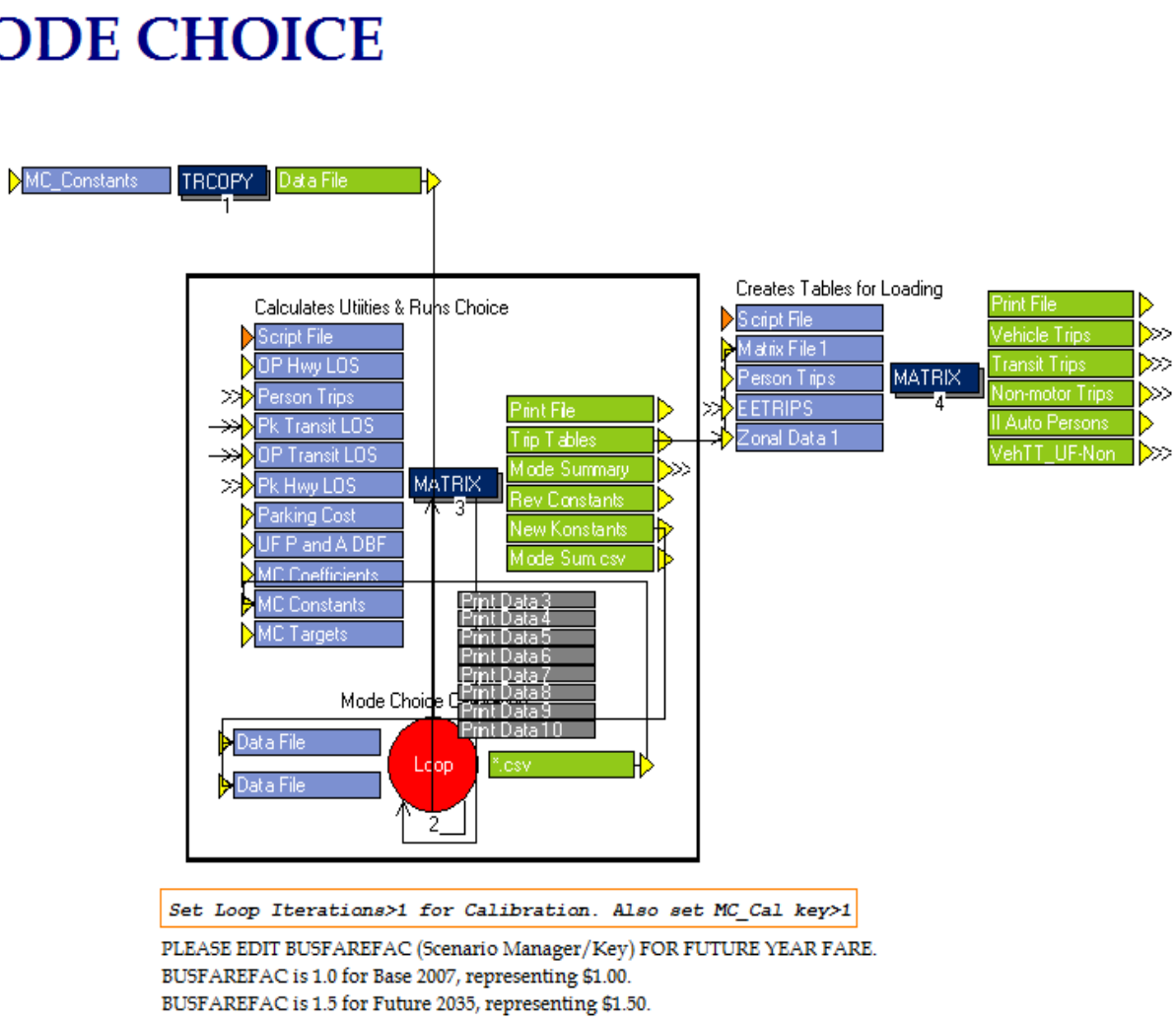
The Federal Transit Administration has noted that certain common practices in transit path building can have undesired impacts on ridership forecasts. Minimum and maximum values of time and distance used to determine valid transit paths and modal availability can have unexpected effects. Path building parameters and settings should remain the same for all steps of the model (skimming, assignment). All transit parameters in the model were reviewed for consistency across the model chain. The careful review of all transit parameters ensured reasonableness of transit access and path building.

Average weekday transit trips in the year 2015 computed from the data provided by the Gainesville Regional Transit System were 48,555. The model currently is estimating an average of 49,125 transit riders per day, which is only 1.1 percent higher than observed data and is well within the acceptable range of three to nine percent from the Florida Standard Urban Transportation Structure standards. Additional details on the transit assignment are provided in Section 8.0 of this report.

6. Mode Choice

Since Alachua County is the home of the University of Florida and Santa Fe College, like other college towns, the transit system plays an important role in daily transportation and needs to be considered in the modeling. The mode choice model used in the 2015 Gainesville Urbanized Area Transportation Study Model is a set of nested logit models that estimate modal shares among several categories of automobile and transit modes. This section of the report describes the structure and validation of the mode choice model. Figure 6.1 shows the structure of this part of the model

Figure 6.1 Mode Choice Model Chain



A.DBF = Attractions. File Format.dbf
EETRIPS = External Trips
LOS = Level of Service
MC = Mode Choice
OP – Off Peak Period
PK = Peak Period
TT = Transit Trips
UF = University of Florida
UFP = University of Florida Productions

6.1 Mode Choice Model Structure

The Florida Standard Urban Transportation Model Structure process makes use of a nested logit model for mode choice, except in the case of the highway only models (i.e., those that do not include transit networks). The entire mode choice process is executed via Cube/Voyager scripting, using the MATRIX program.

One script, MCMAT00A.S, creates a trip table matrix set containing trips for all five purposes: home-based work, home-based other, non-home-based, home-based university and University of Florida Campus/Dormitory. The trip tables for each purpose are separated into eight mode-specific tables: automobile drive-alone, automobile 2-

person carpool, automobile 3+ person carpool, walk to local bus, walk to premium bus, drive to transit, walk, and bicycle. Transit skim data compiled in the transit network module and highway skims generated during trip distribution are inputted into the "MODE" module. The peak period utilizes the congested highway skims, whereas the off-peak period uses the free flow skims.

After running the mode choice model, the outputs are balanced into origin and destination trip tables. A separate script, MCMAT00C.S, combines trip purposes and outputs separate trip tables for the following modes and these trip tables are then used during the highway and transit assignment phases of the model:

1. Drive alone auto;
2. Carpool auto;
3. Light duty trucks;
4. Heavy duty trucks;
5. External-external trips;
6. Peak period transit;
7. Off-peak period transit;
8. Non-motorized travelers; and
9. Internal auto persons (combination of modes 1 and 2, above).

The scripts referenced above and others pertaining to Mode Choice can be found in Appendix D, along with the corresponding model flowchart for each step.

6.2 Development and Validation of Mode Choice Model

The mode choice model step for the 2015 Gainesville Urbanized Area Transportation Study Model was developed from the 2010 Gainesville Urbanized Area Transportation Study Model, with a few changes to mode choice scripting (MCMAT00A.S). The constants file (MCCONSTANTS.CSV) was modified with the recalibration of the transit trips. Coefficients (MCCOEFFICIENTS.CSV) are kept the same as 2010 and are in a numeric range consistent with current Florida Standard Urban Transportation Model Structure validation standards. The mode choice coefficients were not revised, given that a better source of data, such as a new household travel survey, is not available. All parameters are shown in Table 6.1.

Table 6.1 Mode Choice Coefficients for 2015 Model

Mode Choice Model Parameters	2015	FSUTMS
Home-Based Work IVTT	-0.025	-0.01 to -0.05
Home-Based Non-Work IVTT	-0.02	-0.007 to -0.033
Non-Home Based IVTT	-0.024	-0.01 to -0.05
University IVTT	-0.024	-0.02 to -0.03
Home-Based Work OVTT	-0.049	NA
Home-Based Non-Work OVTT	-0.048	NA
Non Home-Based OVTT	-0.07	NA
University OVTT	-0.048	NA
Home-Based Work OVTT/IVTT	2	1.5 to 3.0
Home-Based Non-Work OVTT/IVTT	2.4	2.0 to 6.0
Non Home-Based OVTT/IVTT	2.9	2.0 to 7.0
University OVTT/IVTT	2	2.0 to 3.0

**IVTT = in-vehicle travel time, OVTT = out-vehicle travel time*

6.3 Mode Choice Model Results

Extensive changes were made to the mode choice targets of the home-based university trips. These targets were developed based on the University of Florida Campus Master Plan existing conditions report prepared by the University of Florida/VHB. The availability of this data allowed developing more confident student mode choice targets. The model choice model was recalibrated to match with high non-motorized, transit and carpooling percentages. A big shift in trip mode to transit and non-motorized has been achieved, and which correctly represents the existing conditions.

Table 6.2 contains the 2015 Gainesville Urbanized Area Transportation Study Model Mode Choice validation results. Mode choice results were compared with statistics from Census American Community Survey data for the home-based work trip purpose. The 2015 results are reasonable compared to the transit mode share estimate from the American Community Survey / National Household Travel Survey data (refer to the end of Table 6.2). In total, 2.74 percent of all trips in the 2015 model was allocated to transit modes while 2.38 percent of all trips in the 2010 model was allocated to transit.

Table 6.2 Mode Choice Validation Summary

	2015		2010		2007		
	Trips	Percent	Trips	Percent	Trips	Percent	FSUTMS*
Home-Based Work							
Drive Alone	132,776	77.95%	145,470	78.25%	143,880	78.51%	70-85%
Two Passengers	14,067	8.26%	16,609	8.93%	15,951	8.70%	9-16%
3+ Passengers	7,369	4.33%	8,287	4.46%	8,019	4.38%	2-6%
Total Transit	9,317	5.47%	7,779	4.18%	8,043	4.39%	0.5-10%
Walk	3,716	2.18%	4,339	2.33%	4,098	2.24%	NA
Bicycle	3,098	1.82%	3,412	1.84%	3,278	1.79%	NA
Total	170,343	100.00%	185,896	100.00%	183,269	100.00%	NA
Home-Based Other							
Drive Alone	202,973	40.89%	204,045	38.71%	200,318	39.63%	NA
Two Passengers	187,858	37.84%	201,574	38.24%	191,683	37.92%	NA
3+ Passengers	84,858	17.09%	98,476	18.68%	90,914	17.99%	NA
Total Transit	1,445	0.29%	1,852	0.35%	2,943	0.58%	NA
Walk	17,773	3.58%	19,425	3.68%	17,952	3.55%	NA
Bicycle	1,511	0.30%	1,786	0.34%	1,631	0.32%	NA
Total	496,418	100.00%	527,158	100.00%	505,441	100.00%	NA
Non Home-Based							
Drive Alone	187,939	54.77%	172,087	48.61%	166,902	50.70%	NA
Two Passengers	106,858	31.14%	117,608	33.22%	106,625	32.39%	NA
3+ Passengers	41,386	12.06%	52,509	14.83%	45,341	13.77%	NA
Total Transit	1,324	0.39%	2,415	0.68%	2,587	0.79%	NA
Walk	3,219	0.94%	6,274	1.77%	5,105	1.55%	NA
Bicycle	2,441	0.71%	3,158	0.89%	2,640	0.80%	NA
Total	343,167	100.00%	354,051	100.00%	329,200	100.00%	NA
Home-Based University							
Drive Alone	13,907	19.12%	32,169	49.93%	34,900	45.96%	NA
Two Passengers	4,983	6.85%	3,915	6.08%	4,399	5.79%	NA
3+ Passengers	3,079	4.23%	1,960	3.04%	2,261	2.98%	NA
Total Transit	17,224	23.68%	13,478	20.92%	9,178	12.09%	NA
Walk	15,849	21.79%	6,811	10.57%	15,025	19.78%	NA
Bicycle	17,704	24.34%	6,090	9.45%	10,179	13.40%	NA
Total	72,746	100.00%	64,423	100.00%	75,942	100.00%	NA
University of Florida Campus/Dorm							
Total Transit	1,709	7.19%	1,947	8.55%	1,981	8.40%	NA
Walk	15,785	66.40%	15,180	66.68%	15,615	66.25%	NA
Bicycle	6,278	26.41%	5,641	24.78%	5,974	25.35%	NA
Total	23,772	100.00%	22,767	100.00%	23,570	100.00%	NA
All Purposes							
Drive Alone	537,595	48.59%	553,771	47.97%	546,000	47.80%	35-50%
Two Passengers	313,766	28.36%	339,706	29.43%	318,658	27.90%	15-35%

	2015		2010		2007		
	Trips	Percent	Trips	Percent	Trips	Percent	FSUTMS*
Home-Based Work							
3+ Passengers	136,692	12.35%	161,232	13.97%	146,535	12.83%	8-20%
Total Transit	31,019	2.80%	27,471	2.38%	24,732	2.17%	0.2-9%
Walk	56,342	5.09%	52,029	4.51%	57,795	5.06%	NA
Bicycle	31,032	2.80%	20,087	1.74%	23,702	2.08%	NA
Total	1,106,446	100.00%	1,154,295	100.00%	1,142,154	100.00%	NA

Table 6.3 Mode-Choice American Community Survey Comparison

Classes	American Community Survey 2014-2018 Five-Year Estimates		CTPP/NHTS Five-Year Estimates	
	Estimate	Percent	Estimate	Percent
Drove alone	91,254	75%	86,245	75%
Carpooled	11,452	9%	10,165	9%
Transit	4,705	4%	4,915	4%
Bicycle	2,824	2%	3,125	3%
Walked	3,538	3%	3,560	3%
Taxicab, motorcycle, or other means	2,063	2%	2,154	2%
Worked at home	5,619	5%	5,295	5%
Total	121,455	100%	115,459	100%

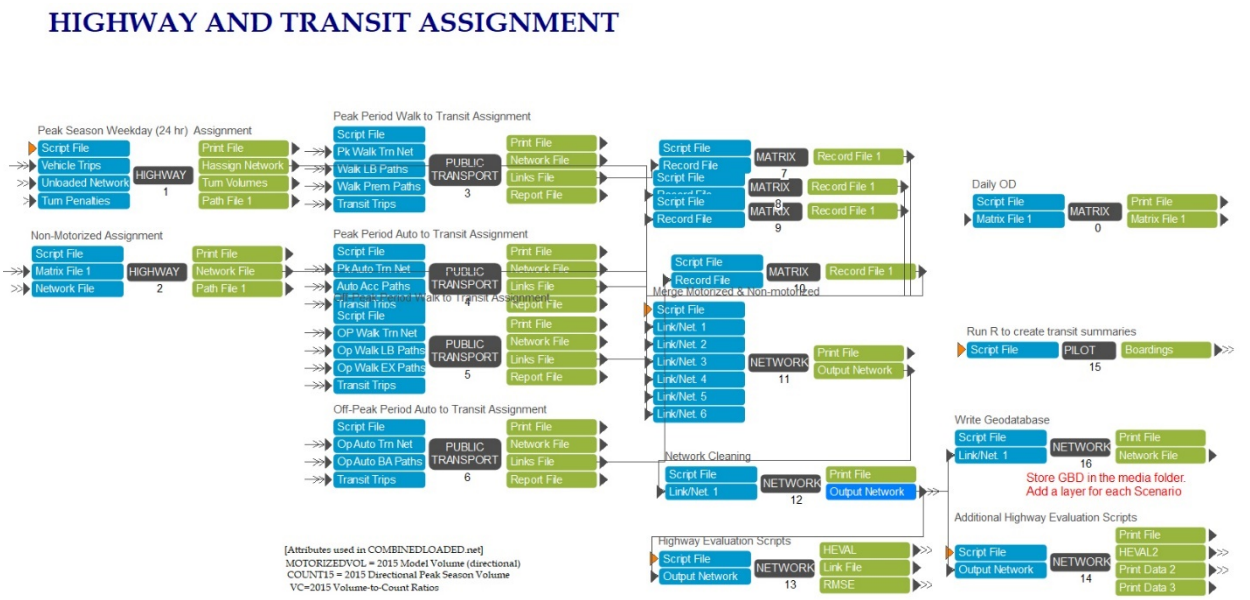
CTPP = Census Transportation Planning Products
NHTS = National Household Travel Survey

7. Highway Assignment

Proper validation of the highway assignment is critical to the meaningful application of travel demand models. The objective of the assignment module is to load trips to available paths to achieve system equilibrium. Highway trip assignment determines the shortest path between any given origin-destination pair and loads the automobile trips produced by the mode choice module onto those path links. The trips are loaded incrementally – once the first iteration is completed, link travel times are then recomputed, the new shortest paths are subsequently determined, and the next iteration is executed.

This process proceeds through a series of iterative calculations until no one can shorten their travel time by changing routes. In other words, system equilibrium is achieved at the end of the highway assignment. The red dotted line in Figure 7.1 shows the highway assignment model chain. The highway assignment is combined with the transit assignment at this model. The details of the transit assignment will be discussed in the next section.

Figure 7.1 Highway Assignment Module



7.1 Highway Assignment Procedure

Automobile trips are loaded onto the network utilizing an iterative equilibrium highway load program based on an all or nothing assignment algorithm. In total, six model runs were executed to validate the 2015 Gainesville Urbanized Area Transportation Study Model, excluding numerous test-runs that were necessary for iterative adjustment and calibration. Adjustments were made to key elements of the modeling process to achieve satisfactory validation results. After each run, a summary of the results was compiled and analyzed to identify areas for improvement in the model and successful strategies toward validation enhancement.

A few key evaluation statistics, which are recommended by the Florida Standard Urban Transportation Model Structure-Cube Framework Phase II: Model Calibration and Validation Standards Final Report, were generated for model calibration and validation purposes. Model validation was accomplished by minimizing the difference between the model estimated volumes and observed traffic counts for the year 2015 on network segments for the entire study area.

Extensive changes to the Highway Assignment module were made as part of the 2015 calibration. Significant improvements to the equilibrium assignment closure criteria were made using the CUBE's latest highway assignment functions. The assignment iterations were increased to 100 and the "relative gap" closing criteria of 0.0001 was used. This will ensure the highway assignment runs enough iterations and provides "stable" forecasts. To facilitate this process, the "converge" phase was added.

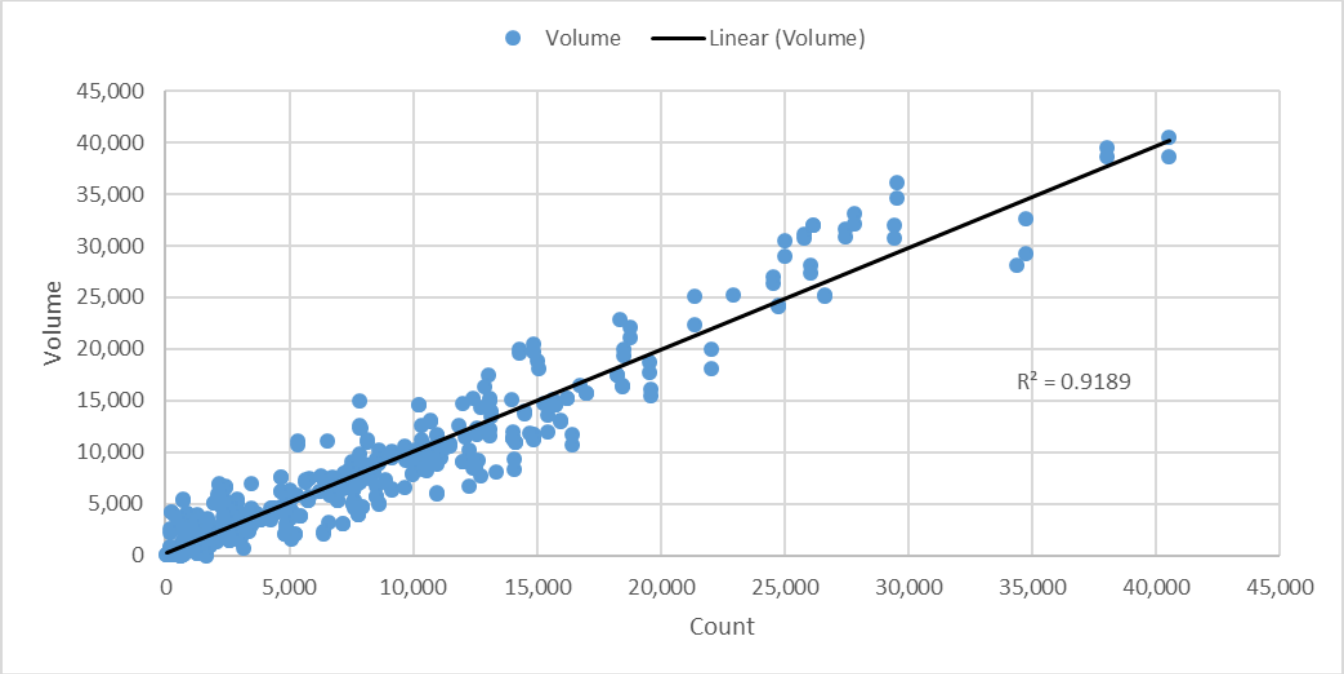
Furthermore, an attempt was made to apply separate volume-delay functions by different facility types. This process did not result in any better than the original single volume-delay function in the model. However, it is a best practice to have the volume-

delay functions sensitive to the network facility types. This holds a place in the model, however, currently all facility types use a single volume-delay function.

7.2 Highway Assignment Validation Results

The evaluation of the highway assignment module is based on comparisons between observed traffic counts and model estimated volumes. Base year model estimated traffic volumes are compared to the 2015 traffic counts in different ways to determine whether the model projections are reasonable. Figure 7.2 shows a volume to count scatter plot for the overall model. The indicators that are available for determining the overall performance of the highway assignment model are volume to count ratios by area type/facility type/lane categories, screen line volume to count ratios, and root mean square error. Each of the measures is discussed in the remainder of this section.

Figure 7.2 Volume to Count Scatter Plot



Assignment Performance by Area Type/Facility Type/Lanes Categories

Table 7.1 contains details of the volume to count ratios for each link group category by area type, facility type, and number of lanes. At the area-wide level, the 2015 base year model volume to count ratios matches well.

Screen Line Performance

Analyzing volume to count ratios along screen lines allows for examining traffic flows across geographic subareas and corridors. Florida Standard Urban Transportation Model Structure along screen lines have historically varied from +/-5 percent to +/-20 percent. There are ten screen lines in the 2015 Gainesville Urbanized Area Transportation Study Model including an external cordon line, measuring trips coming into and going out of the study area.

Table 7.2 shows volume to count ratios by screen line (note that double counting of trips was avoided in creating the summary). Figure 7.3 shows the locations of the screen lines. Screen line 4 and screen line 7 does fall slightly outside of the Florida Standard Urban Transportation Model Structure standard. However, this validation effort was focused on the overall region wide calibration.

Table 7.1 Volume to Count Performance by Category

Category	2015	Florida Standard Urban Transportation Model Structure		Number of Links	
Facility Type		Acceptable	Preferable	Total	with Counts
Freeway	0.98	+-7%	+-6%	44	16
Divided Arterial	1	+-15%	+-10%	1,263	254
Undivided Arterial	1.02	+-15%	+-10%	738	110
Collectors	1.12	+-25%	+-20%	2,781	308
One-Way/Frontage	1.11	+-25%	+-20%	36	4
Area Type					
Central Business District	1.39	NA	NA	212	33
Central Business District Fringe	1.08	NA	NA	342	49
Residential	0.99	NA	NA	1,677	238
Other Business District	1.00	NA	NA	1,004	162
Rural	1.02	NA	NA	1,727	242
Number of Lanes					
One Lane	1.08	NA	NA	3,479	428
Two Lanes	0.98	NA	NA	1,310	247
Three Lanes	1.03	NA	NA	173	47
Total	1.02	+-5%	NA	4,962	724

Table 7.2 Volume to Count Performance by Screen line

Screenline	2015 Volumes Over Counts			Florida Department of Transportation Criteria*	Screenline Description
	Total Volume	Total Count	V/C Ratio		
1	172,837	147,624	1.17	+/-20%	Crossing West of I-75
2	91,530	84,762	1.08	+/-15%	Crossing East University of Florida Campus
3	125,236	134,094	0.93	+/-15%	Crossing State Road 121
4	29,587	24,356	1.21	+/-20%	East-West Cutline West of I-75
5	137,773	129,024	1.07	+/-10%	North-South Crossing State Road 222 (39th Avenue)
6	97,696	91,784	1.06	+/-10%	North-South Cutline in Northwest County/High Springs
7	12,914	9,962	1.30	+/-20%	La Crosse Area
8	34,978	34,467	1.01	+/-20%	East-West Crossing U.S. 301
9	106,889	121,682	0.88	+/-15%	Micanopy Area
10	229,417	230,477	1.00	+/-1%	External Cordon
11	191,357	193,526	0.99	+/-20%	Crossing East of I-75
ALL	1,230,214	1,201,759	0.99	+/-15%	Overall

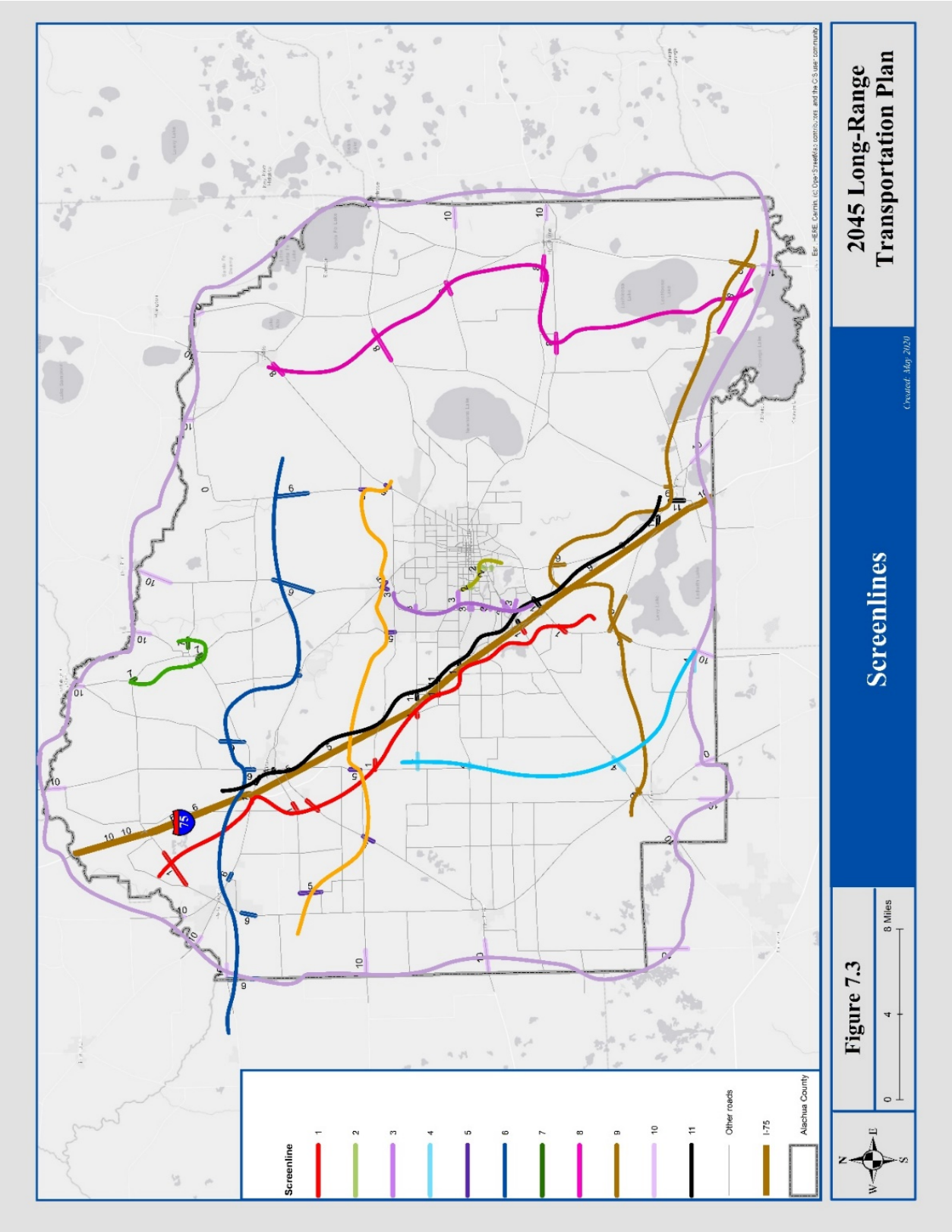
Percent Root Mean Square Error

Root Mean Square Error is the commonly reported highway evaluation statistics in model validation. Root Mean Square Error, a measure of dispersion, tends to normalize model error better than volume to count ratios that allow for high ratios to offset low ratios. Table 7.3 shows the Root Mean Square Error statistics by volume group. All categories met established accuracy standards.

Table 7.3 Root Mean Squared Error

Volume Range	Number of Links	2015 Model	Florida Standard Urban Transportation Model Structure	
			Acceptable	Preferable
1-5,000	369	77.10%	+/-100%	+/-45%
5,000-10,000	172	29.70%	+/-45%	+/-35%
10,000-20,000	145	20.20%	+/-30%	+/-25%
20,000-30,000	30	14.00%	+/-27%	+/-15%
30,000-40,000	6	13.50%	+/-25%	+/-15%
40,000- 50,000	2	4.70%	+/-25%	+/-15%
Average Total	724	30.30%	+/-45%	+/-35%

Figure 7.3 Screenlines



8. Transit Procedure and Assignment

The transit procedure in the Gainesville Urbanized Area Transportation Study 2015 model includes the development of transit input data, building transit path, calculating the number of transit trips from the mode choice model, and loading the transit trips to the transit network. Each of these four aspects of the transit procedure is discussed in this document.

The development of the transit input data was discussed in Section 5. Information on routes, stop locations, service characteristics, and ridership was provided by the City of Gainesville Regional Transit System staff. Detail transit network coding and review were conducted in order to ensure that the system representation in the model is accurate. The ridership data was utilized to assess the reasonableness of the model outputs. A detailed review of all transit parameters including path building was conducted and documented.

Transit levels of service are computed separately for peak hours and off-peak hours. During the process of distribution, the preloaded highway network with pre-assignment loads based on initial time skim is created. Then, in the process of transit path creation, the preloaded network is used to calculate the time skim for transit during peak hours. Transit vehicle speed (link travel time) is determined as a function of the automobile speed on each link. The type of relationship used for each transit service type is identified based on area type and facility type of the link. These automobile/transit speed relationships are used in a lookup function to assign speeds for transit paths.

Then the transit travel time is calculated proportionally according to the relative relationship between automobile travel times. During this step, path files with shortest time and distance for peak and off-peak are also created which are later used at the transit assignment stage. Other variables, like headway, transit accessibility, waiting time, and transit path building parameters are assigned to each route during this process as well.

The Gainesville Urbanized Area Transportation Study 2015 model uses a nested logit model approach for mode choice. Variables that are considered include in-vehicle transit time, out-of-vehicle transit time, cost, bike or walk to transit time, and pedestrian environment variables. The coefficients used in mode choice for these variables are shown in Table 8.1. For home-based work trips, the local bus fare is discounted to 25 percent since there is an employee pass program. This 25 percent factor was retained from the 2010 model and was originally computed to match model computed home-based work ridership to observed ridership. Since the transit fare is covered by tuition fees of University of Florida students, the local bus fare for home-based university and dorm-university walk to local transit is discounted to 10 percent. For home-based university walk and drive to premium transit, the bus fare is free. The percentages of each mode are calculated by using the nested logit mode-choice. The details of the mode choice model validation are discussed in Section 6.

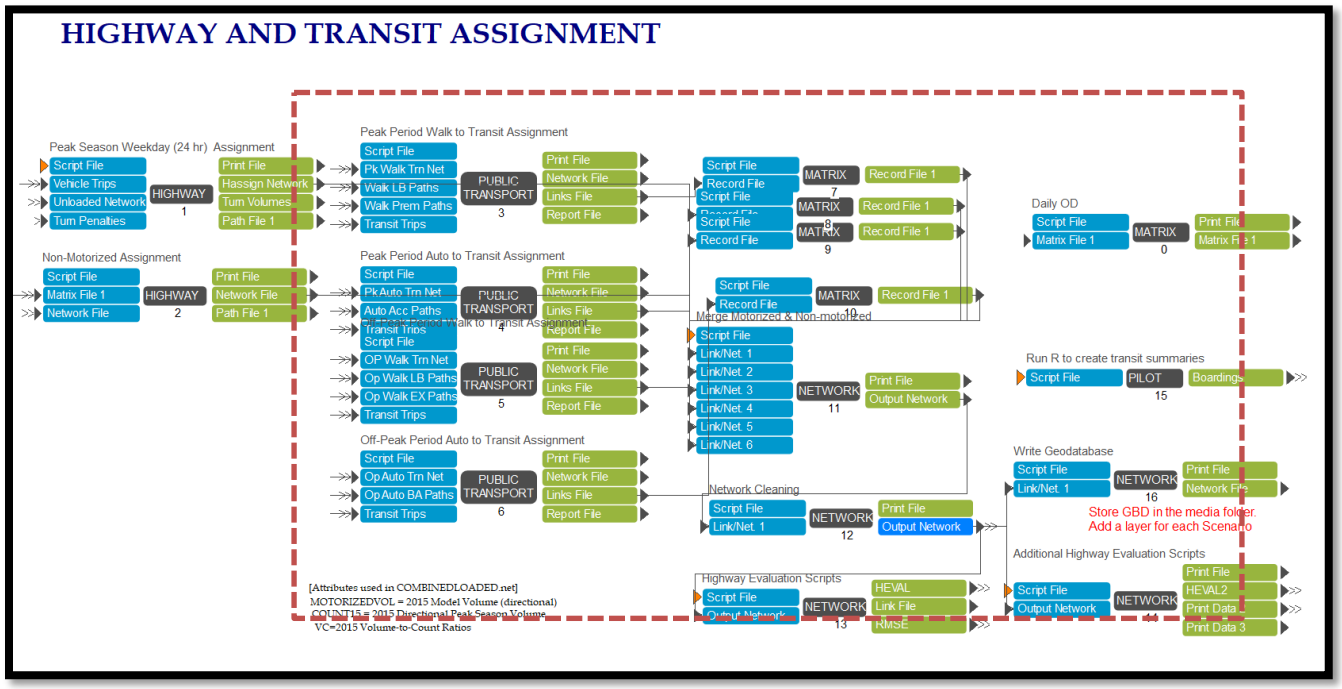
Table 8.1 Mode Choice Coefficient for Transit Variables

Coefficient	Home Based Work	Home Based Other	Non-Home Based	University
civt - In-Vehicle Travel Time Coefficient	-0.025	-0.02	-0.024	-0.024
covt – Out-of-Vehicle Travel Time Coefficient	-0.049	-0.048	-0.07	-0.048
ccst - Cost Coefficient	-0.005	-0.011	-0.009	-0.011
cwt - Walk only Coefficient	-0.042	-0.083	-0.052	-0.083
cbt - Bike Coefficient	-0.109	-0.117	-0.096	-0.117
pti - Walk to Transit Pedestrian Environment Variable Coefficient	1.15	0.6	0.45	0.25
pwi - Walk Pedestrian Environment Variable Coefficient Origin	0.35	0.175	0.22	0.4
pwj - Walk Pedestrian Environment Variable Coefficient Destination	0.3	0.164	0.164	0.35
pbi - Bike Pedestrian Environment Variable Coefficient Origin	0.47	0.07	0.066	0.3
pbj - Bike Pedestrian Environment Variable Coefficient Destination	0.006	0	0.006	0.006

cbt = Bike Coefficient
ccst = Cost Coefficient
civt = In-Vehicle Travel Time Coefficient
covt = Out-of-Vehicle Travel Time Coefficient
cwt = Walk only Coefficient
pbi = Bike Pedestrian Environment Variable Coefficient Origin
pbj = Bike Pedestrian Environment Variable Coefficient Destination
pti = Walk to Transit Pedestrian Environment Variable Coefficient
pwi = Walk Pedestrian Environment Variable Coefficient Origin
pwj = Walk Pedestrian Environment Variable Coefficient Destination

The last step in the process is to assign transit trips to the routes. Separate loads are conducted by mode and period as allocated in the mode choice model. Home-based work trips are assigned to the peak period network while home-based other, non-home-based, home-based university and dorm-university trips are assigned to the off-peak network. Within each period, there are three transit mode choices available: walk to local bus, walk to the express bus, and drive to the best available transit. The transit assignment flow is relatively simple in this model. It loads the transit trips from mode choice onto the transit paths computed during path building. There are no parameters used in this procedure. The red dotted square in Figure 8.1 shows the transit assignment module.

Figure 8.1 Transit Assignment Module



8.1 Transit Assignment Validation Results

Not nearly as much effort is typically expended on transit assignment validation as is devoted to highway assignment validation, in part due to transit assignment not impacting roadway congestion or capacity, as well as a general lack of post-processing programs and scripts to summarize transit assignment accuracy. However, transit is a key component of the Gainesville Urbanized Area Transportation Study 2015 model and it is important to validate the transit assignment reasonably well at the system level at a minimum. The transit assignment step in the model chain loads person trips onto the transit network. Separate loadings are conducted by mode and period: home-based work trips are assigned to the peak period network while home-based other, non-home-based, home-based university and dorm-university trips are assigned to the off-peak network.

Overall validation for transit assignment was based on an analysis of the transit ridership not only as a system, but also on a route by route basis. Transit assignment loadings were compared to average weekday "unlinked" route ridership data provided by the City of Gainesville Regional Transit System. Unlinked trips are equal to bus boarding and count all the times that a person boards a bus, both initially and during transfers. Table 8.2 shows the route level comparison of observed ridership and ridership estimated by the model. The 2015 City of Gainesville Regional Transit System weekday ridership was calculated using the average of the months when school is in session during 2015 (excluding summer months of June, July, and August). The average weekday ridership during the summer months is much lower than the rest of the year. Since the model forecasts include university trips, it is reasonable to use average weekday ridership from September to May for comparison purposes.

The Gainesville Urbanized Area Transportation Study 2015 model trips are the number of transit trips loaded to each route by the transit assignment module. System-wide model estimates of unlinked trips are very close to City of Gainesville Regional Transit System observed numbers. As shown in Table 8.2, the 2015 transit assignment model estimated 49,612 "unlinked" riders system-wide, while the Gainesville Regional Transit System reported 48,555 average weekday ridership system-wide for 2015 while school was in session.

At the system-wide level, ridership for the Gainesville Urbanized Area Transportation Study 2015 model differs by only two percent from data provided by Gainesville Regional Transit System. This is within the preferred three to nine percent Florida Department of Transportation validation standard. In general, regional models do not validate well on a route by route basis. The Gainesville Urbanized Area Transportation Study 2015 model performs well in transit assignment at the system-wide level and performs reasonably well on a route-by-route basis.

Appendix D contains the model script files and flowchart for the transit assignment. These show the processes, input parameters, and output file locations.

Table 8.2 Transit Loading Estimates: Year 2015

Route	Route Name	2015 Model Volume	2015 Average Weekday Ridership (RTS)	Validation
1	Downtown Station to Butler Plaza	3,828	2,678	1.41
2	Downtown Station to NE Walmart Supercenter	164	315	0.49
2B	Downtown Station to N Main Street Post Office	159	113	1.36
5	Downtown Station to Oaks Mall	2,718	1,856	1.45
6	Downtown Station to Plaza Verde	358	341	1.00
7	Downtown Station to Eastwood Meadows	128	348	0.36
8	Shands to North Walmart Supercenter	1,548	1,245	1.14
9	Reitz Union to Hunters Run	994	3,017	0.33
10	Downtown Station to Santa Fe	873	575	1.46
11	Downtown Station to Eastwood Meadows	302	568	0.51
12	Reitz Union to Butler Plaza	2,553	3,334	0.76
13	Beaty Towers to CareerSource	745	1,554	0.47
15	Downtown Station to NW 13th Street (@ NW 23rd Avenue)	745	1,003	0.73
16	Beaty Towers to Sugar Hill	684	542	1.24
17	Beaty Towers to Downtown Station	707	702	1.02
20	Reitz Union to Oaks Mall	4,725	5,344	0.89
21	Reitz Union To Cabana Beach	3,137	2,239	1.40
38T	The Hub to Old Archer Rd	88	63	1.32
23	Oaks Mall to Santa Fe	325	863	0.37
24	Downtown Station to Job Corps	247	274	0.87
25	University of Florida Commuter Lot to Airport	503	321	1.49
24B	Downtown Station to Airport	228	234	1.00
27	Downtown Station to NE Walmart Supercenter	265	98	2.66
28	The Hub to Forest Park	882	778	1.15
34	The HUB to Lexington Crossing	1,930	1,481	1.29
35	Reitz Union to SW 35th Place	2,246	2,455	0.88
36	Reitz Union to SW 34th Street Post Office	697	527	1.22
38	The Hub to Gainesville Place	2,641	3,028	0.86
39	Santa Fe to Airport	144	131	1.18
41	Beaty Towers to North Walmart Supercenter	1,311	334	3.85
43	Shand to Santa Fe	1,537	1,046	1.47
46	Reitz Union to Downtown Station	656	772	0.78
62	Oaks Mall to Lexington Crossing	311	93	3.32
75	Oaks Mall to Butler Plaza	458	944	0.46
76	Santa Fe to Haile Square Market	283	206	1.36
77	Santa Fe to Cabana Beach Apts	69	109	0.64
117	Park-N-Ride 2 (SW 34th Street)	564	1,380	0.40
118	Park-N-Ride 1 (Cultural Plaza)	1,691	2,145	0.79
119	Family Housing	283	304	0.93

Route	Route Name	2015 Model Volume	2015 Average Weekday Ridership (RTS)	Validation
120	West Circulator (Fraternity Row)	2,093	1,453	1.45
121	Commuter Lot	683	429	1.49
122	University of Florida North/South Circulator	725	213	3.36
125	Lakeside	2,820	1,527	1.83
126	University of Florida East/West Circulator	1,522	494	3.06
127	East Circulator (Sorority Row)	288	1,092	0.26
Systemwide		49,125	48,555	1.01

N = North
NE = Northeast
NW = Northwest
RTS = Regional Transit System
SW = Southwest

9. Summary and Conclusions

The model validation phase of the Gainesville Urbanized Area Year 2045 Long-Range Transportation Plan Update began with data development and review which was documented in Technical Reports 2 and 3. Data review, adjustment, and correction were an iterative process throughout the model validation effort, reflecting identification of data issues based on model results.

Once the input data were acceptable, work continued on validating each component of the Gainesville Urbanized Area Transportation Study 2015 travel demand model. The highway side of the model validated reasonably well from the start, largely a reflection of the efforts put forth by the study team in data development, review, disaggregation, and refinement. The transit portion of the model validation is acceptable for system wide performance evaluations. For route-level ridership forecasting studies, additional corridor validation is recommended.

The Gainesville Urbanized Area Transportation Study 2015 base year model meets most established Florida Standard Urban Transportation Model Structure standards for model accuracy and reasonableness. The validated model will be used in subsequent phases of the Gainesville Urbanized Area Year 2045 Long-Range Transportation Plan Update to develop and test transportation alternatives.

9.1 Model Usability Improvements and Dashboard

The following extensive improvements to the reporting of model statistics have been achieved.

1. The 2015 model now generates Screenline summary in the RMSE.prn file.
2. The model generates Transit Boarding Statistics.xlsx (STEP6 Assignment-> STEP15) that summarizes the transit ridership data by routes, modes and stops in separate tabs. This is a product of specialized software using R. The R software files are also provided to the user in the 2015 Gainesville Urbanized Area Transportation Study

travel demand model distribution package called R-setup. These are automatically called during the model run without user intervention.

3. The model also generated a reporting dashboard in HTML format (STEP7- right click and open in Windows). This can be opened in any internet browser.

Appendix A: Socio-economic Data Format

ZONEDATA{YEAR}.DBF

Notes: Please note that ZONEDATA{YEAR}.DBF is part of geographic information systems traffic analysis zone shape database of ZONEDATA.SHP, together with ZONEDATA.SHX. When editing this ZoneData file, Cube or geographic information systems software must be used.

Attribute List for Population and Household Data

TAZ_2015 – Traffic Analysis Zone number in the 2015 Gainesville Urbanized Area Transportation Study model.
TAZ_2010 – Traffic Analysis Zone number in the 2010 Gainesville Urbanized Area Transportation Study model.
SFDU – Number of single-family dwellings units
SPOP – Population in single-family dwellings units
MFDU – Number of multifamily dwellings units
MPOP – Population in multifamily dwellings units
TOTPOP15 – Total population for year 2015 (this attribute is not used by model scripts, instead population in single-family dwelling units and population in multi-family dwelling units are used for base year and future year scenarios)
HMDU – Total hotel–motel units
HMPOP – Total population in occupied hotel-motel units

Attribute List for Employment Data

OIEMP – Other industrial employment
MFGEMP – Manufacturing industrial employment
COMEMP – Commercial employment
SERVEMP – Service employment (includes University of Florida employment)
TOTEMP – Total employment (includes University of Florida employment)
SCHENR – School enrollment by school location (this excludes any University of Florida or Santa Fe College enrollment)

Attribute List for University of Florida Data

UF_EMP – Number of University of Florida place-of-work employees by traffic analysis zone (this variable also is used to reallocate service employment on University of Florida Campus)
UF_DORM_ST – Number of on-campus University of Florida student residents
UF_PARKING – University of Florida commuting parking spaces, excluding on-campus student long-term
UF_ST_PARK – University of Florida commuting student parking spaces

CLASSROOMS – Number of University of Florida classrooms (model scripts do not use this variable)
CLASSSQFT – Square feet of University of Florida classrooms (model scripts do not use this variable)
SEATS – Number of University of Florida classroom seats
UF-OC-ST – Number of University of Florida off-campus students, estimated from student address records provided by University of Florida
SUB_AREA – Name of city or incorporated area or Alachua County if a zone is within the unincorporated area
UFZONES – Identifier that indicates that a zone is on University of Florida Campus when the value is one

Attribute List for Transit PEV (Pedestrian Environment Variable) Data
SIDEWALK – Sidewalk availability (values vary from 0 to 3)
CROSSING – Ease of street crossing (values vary from 0 to 3)
NONMTR_CNN – Non-motorized connections (values vary from 0 to 3)
SETBACK – Building setbacks (values vary from 0 to 3)
SUM – Sum of four variable values above: SIDEWALK, CROSSING, NONMTR_CNN and SETBACK (SUM needs to be updated manually when any of four variables has been modified)
COMPOSIT – composite pedestrian environment variable value (model scripts do not use this)
SELECTZONE – Identifier that indicates that a zone is selected for select zone analysis when the value is one (the model will load selected trips that end at the selected zones, and it will be reported in the attribute SELZONE_MOTOR in the final highway assignment output network, COMBINEDLOADED.NET)
HOTEL – Identifier used in the previous model (model scripts do not use this variable)

Attribute List for Parking Data (same as 2010)
SHORTPARK – Short-term (three hour) parking cost (cents)
LONGPARK – Long-term (eight hour) parking cost (cents)
STUDENTPAR – Student (eight hour) parking cost (cents) at University of Florida

Attribute List for Population and Household Variable Data (same as 2010)
SF_SEA – Percent single family dwelling unit not occupied by permanent residents
SF_0V – Percent households having no vehicles in single-family dwelling unit occupied by permanent residents
SF_1V – Percent households having one vehicle in single family dwelling unit occupied by permanent residents
SF_2V – Percent households having two vehicles in single family dwelling unit occupied by permanent residents
SF_3V – Percent households having three or more vehicles in single family dwelling unit occupied by permanent residents
SF_VAC – Percent single family dwelling unit vacant
MF_SEA – Percent multi family dwelling unit not occupied by permanent residents

MF_0V – Percent households having no vehicles in multi family dwelling unit occupied by permanent residents
MF_1V – Percent households having one vehicle in multi family dwelling unit occupied by permanent residents
MF_2V – Percent households having two vehicles in multi family dwelling unit occupied by permanent residents
MF_3V – Percent households having three or more vehicles in multi-family dwelling unit occupied by permanent residents
MF_VAC – Percent multi family dwelling unit vacant
HM_POC – Percent hotel-motel units occupied

Appendix B: Turn Penalties (TURN.PEN)

Origin Node A	Intersection Node B	Destination Node C	Penalty Set	Penalty Value*
1211	1207	1221	1	-1
1207	1211	1221	1	-1
1214	1211	1218	1	-1
1214	1211	1221	1	-1
1214	1218	1211	1	-1
1240	1239	1241	1	-1
1240	1241	1238	1	-1
1240	1241	1239	1	-1
1242	1241	1238	1	-1
1241	1242	1238	1	-1
1326	1320	1324	1	-1
1326	1324	1320	1	-1
1326	1324	1325	1	-1
1328	1324	1325	1	-1
1324	1328	1325	1	-1
1338	1333	1337	1	-1
1338	1337	1333	1	-1

Origin Node A	Intersection Node B	Destination Node C	Penalty Set	Penalty Value*
1338	1337	1339	1	-1
1340	1337	1339	1	-1
1337	1340	1339	1	-1
1468	1467	1472	1	-1
1468	1467	1474	1	-1
1468	1467	5404	1	-1
1472	1467	1474	1	-1
1472	1467	5404	1	-1
1467	1472	1474	1	-1
1468	1472	1467	1	-1
1468	1472	1474	1	-1
1485	1484	1483	1	-1
1485	1484	1486	1	-1
1486	1484	1483	1	-1
1484	1486	1483	1	-1
1485	1486	1483	1	-1
1485	1486	1484	1	-1
1588	1581	1585	1	-1
1581	1588	1585	1	-1
1589	1588	1585	1	-1
1589	1588	1593	1	-1
1589	1593	1588	1	-1
1599	1597	1601	1	-1
1599	1601	1597	1	-1
1599	1601	1602	1	-1
1603	1601	1602	1	-1

Origin Node A	Intersection Node B	Destination Node C	Penalty Set	Penalty Value*
1601	1603	1602	1	-1
1737	1733	1740	1	-1
1737	1733	1744	1	-1
1740	1733	1744	1	-1
1733	1740	1744	1	-1
1737	1740	1733	1	-1
1737	1740	1744	1	-1
1752	1750	1749	1	-1
1752	1750	1757	1	-1
1757	1750	1749	1	-1
1750	1757	1749	1	-1
1752	1757	1749	1	-1
1752	1757	1750	1	-1
1828	1825	1830	1	-1
1828	1830	1825	1	-1
1828	1830	1829	1	-1
1831	1830	1829	1	-1
1831	1835	1829	1	-1
1842	1841	1846	1	-1
1842	1846	1841	1	-1
1842	1846	1843	1	-1
1858	1846	1843	1	-1
1846	1858	1843	1	-1
2842	2841	2844	1	-1
2842	2841	2846	1	-1
2844	2841	2846	1	-1

Origin Node A	Intersection Node B	Destination Node C	Penalty Set	Penalty Value*
2841	2844	2846	1	-1
2842	2844	2841	1	-1
2842	2844	2846	1	-1
2858	2856	2855	1	-1
2858	2856	2859	1	-1
2859	2856	2855	1	-1
2856	2859	2857	1	-1
2858	2859	2856	1	-1
2858	2859	2857	1	-1
1472	5356	5409	1	-1

*Penalty value of -1 indicates a movement that is prohibited

Appendix C: Friction Factors

TIME	HBWFF	HBSHFF	HBSRFF	HBOFF	NHBFF	TK4FF	TKSGLFF	TKTRLRFF	SOVIEFF	HOVIEFF	TKLTIEFF	TKHTIEFF	HBUFF	HDORMUFF
1	25208	126687	126687	126687	198262	9231	9048	9704	222	222	222	222	126687	126687
2	21983	47324	47324	47324	71259	8521	8187	9418	333	333	333	333	47324	47324
3	19282	25585	25585	25585	37571	7866	7408	9139	444	444	444	444	25585	25585
4	16953	16092	16092	16092	23174	7261	6703	8869	555	555	555	555	16092	16092
5	14924	10997	10997	10997	15577	6703	6065	8607	666	666	666	666	10997	10997
6	13149	7919	7919	7919	11056	6188	5488	8353	777	777	777	777	7919	7919
7	11591	5913	5913	5913	8147	5712	4966	8106	888	888	888	888	5913	5913
8	10222	4534	4534	4534	6170	5273	4493	7866	1333	1333	1333	1333	4534	4534
9	9018	3548	3548	3548	4773	4868	4066	7634	1666	1666	1666	1666	3548	3548
10	7957	2820	2820	2820	3753	4493	3679	7408	3333	3333	3333	3333	2820	2820
11	7023	2271	2271	2271	2991	4148	3329	7189	6666	6666	6666	6666	2271	2271
12	6199	1849	1849	1849	2410	3829	3012	6977	7777	7777	7777	7777	1849	1849
13	5473	1519	1519	1519	1960	3535	2725	6771	8888	8888	8888	8888	1519	1519
14	4833	1257	1257	1257	1607	3263	2466	6570	9999	9999	9999	9999	1257	1257
15	4267	1047	1047	1047	1326	3012	2231	6376	9999	9999	9999	9999	1047	1047

TIME	HBWFF	HBSHFF	HBSRFF	HBOFF	NHBFF	TK4FF	TKSGLFF	TKTRLRFF	SOVIEFF	HOVIEFF	TKLTIEFF	TKHTIEFF	HBUFF	HDORMUFF
16	3769	877	877	877	1101	2780	2019	6188	9999	9999	9999	9999	877	877
17	3328	739	739	739	919	2567	1827	6005	9999	9999	9999	9999	739	739
18	2940	625	625	625	771	2369	1653	5827	9999	9999	9999	9999	625	625
19	2597	531	531	531	649	2187	1496	5655	9999	9999	9999	9999	531	531
20	2294	452	452	452	548	2019	1353	5488	6666	6666	6666	6666	452	452
21	2026	387	387	387	465	1864	1225	5326	3333	3333	3333	3333	387	387
22	1790	331	331	331	395	1720	1108	5169	1111	1111	1111	1111	331	331
23	1582	285	285	285	337	1588	1003	5016	444	444	444	444	285	285
24	1397	246	246	246	288	1466	907	4868	222	222	222	222	246	246
25	1235	212	212	212	247	1353	821	4724	111	111	111	111	212	212
26	1091	184	184	184	212	1249	743	4584	66	66	66	66	184	184
27	964	159	159	159	183	1153	672	4449	22	22	22	22	159	159
28	852	138	138	138	157	1065	608	4317	16	16	16	16	138	138
29	753	120	120	120	136	983	550	4190	13	13	13	13	120	120
30	665	105	105	105	118	907	498	4066	11	11	11	11	105	105
31	588	92	92	92	102	837	450	3946	16	16	16	16	92	92
32	519	80	80	80	88	773	408	3829	3	3	3	3	80	80
33	459	70	70	70	77	714	369	3716	1	1	1	1	70	70
34	406	61	61	61	67	659	334	3606	1	1	1	1	61	61
35	358	54	54	54	58	608	302	3499	1	1	1	1	54	54
36	317	47	47	47	51	561	273	3396	1	1	1	1	47	47
37	280	41	41	41	44	518	247	3296	1	1	1	1	41	41
38	247	36	36	36	39	478	224	3198	1	1	1	1	36	36
39	219	32	32	32	34	442	202	3104	1	1	1	1	32	32
40	193	28	28	28	29	408	183	3012	1	1	1	1	28	28
41	171	25	25	25	26	376	166	2923	1	1	1	1	25	25
42	151	22	22	22	23	347	150	2837	1	1	1	1	22	22
43	133	19	19	19	20	321	136	2753	1	1	1	1	19	19
44	118	17	17	17	17	296	123	2671	1	1	1	1	17	17
45	104	15	15	15	15	273	111	2592	1	1	1	1	15	15
46	92	13	13	13	13	252	101	2516	1	1	1	1	13	13
47	81	12	12	12	12	233	91	2441	1	1	1	1	12	12
48	72	11	11	11	10	215	82	2369	1	1	1	1	11	11
49	64	9	9	9	9	198	74	2299	1	1	1	1	9	9
50	56	8	8	8	8	183	67	2231	1	1	1	1	8	8
51	50	7	7	7	7	169	61	2165	1	1	1	1	7	7
52	44	7	7	7	6	156	55	2101	1	1	1	1	7	7
53	39	6	6	6	6	144	50	2039	1	1	1	1	6	6
54	34	5	5	5	5	133	45	1979	1	1	1	1	5	5

TIME	HBWFF	HBSHFF	HBSRFF	HBOFF	NHBFF	TK4FF	TKSGLFF	TKTRLRFF	SOVIEFF	HOVIEFF	TKLTIEFF	TKHTIEFF	HBUFF	HDORMUFF
55	30	5	5	5	4	123	41	1920	1	1	1	1	5	5
56	27	4	4	4	4	113	37	1864	1	1	1	1	4	4
57	24	4	4	4	3	105	33	1809	1	1	1	1	4	4
58	21	3	3	3	3	97	30	1755	1	1	1	1	3	3
59	19	3	3	3	3	89	27	1703	1	1	1	1	3	3
60	16	3	3	3	2	82	25	1653	1	1	1	1	3	3
120	0	0	0	0	0	0	0	0	0	0	0	0	0	0

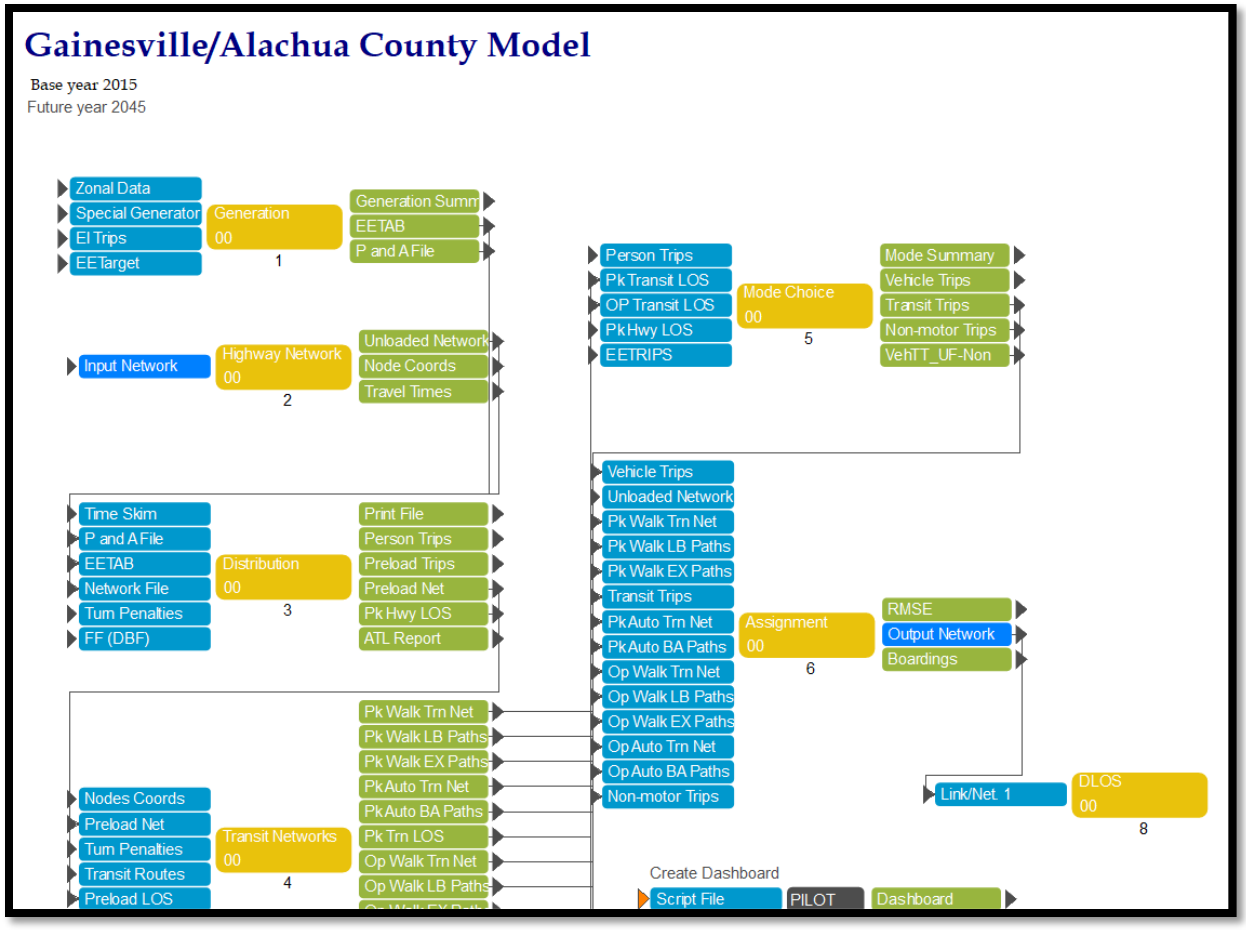
HBOFF – Home Based Other Friction Factor
HBSHFF - Home Based Work *Friction Factor*
HBSR - Home Based Social Recreation *Friction Factor*
HBUFF – Home Based University Friction Factor
HBWFF – Home Based Work Friction Factor
HDORMUFF – Dormitory Based Friction Factor
HOVIEFF - High-occupancy vehicle Friction Factor
NHBFF – None Home Based Friction Factor
SOVIEFF - Single occupant vehicle Internal-External Friction Factor
TK4FF - 4-tire Truck Friction Factor
TKHTIEFF - Heavy Truck Friction Factor
TKLTIEFF - Light Truck Friction Factor
TKSGLFF - Single Unit Truck Friction Factor
TKTRLRFF - Truck Trailer Friction Factor

Appendix D: Model Flowchart, Scripts and File Locations

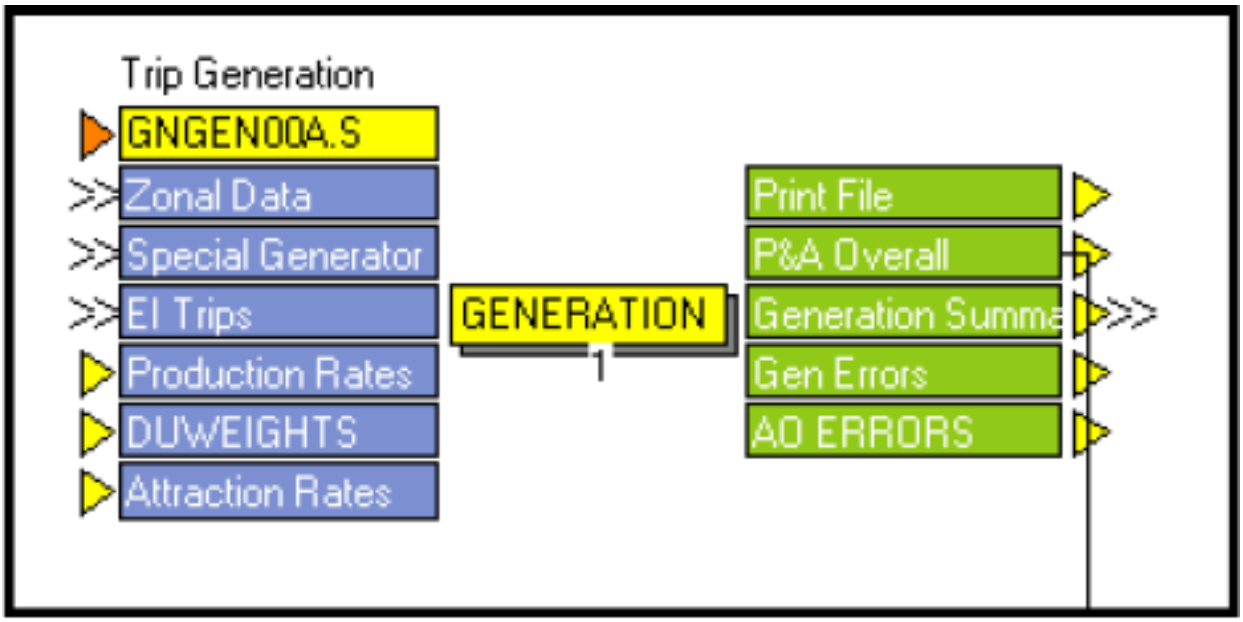
Gainesville 2015 Base Year Model Structure

Model Step	File Name	File Format/ Extension	File Type	Folder Location	Initial Source
Trip Generation	ARATES	DBF	Parameters	\Parameters	Gainesville 2010
	DUWEIGHTS	DBF	Parameters	\Parameters	Gainesville 2010
	EETARGETS	DBF	Input	\Base\Input	FTI, Gainesville 2010
	EITRIPS	DBF	Input	\Base\Input	FTI, Gainesville 2010
	GRATES	DBF	Parameters	\Parameters	Gainesville 2010
	SPGEN	DBF	Input	\Base\Input	Gainesville 2010
	ZONEDATA	DBF	Input	\Base\Input	MTPO Staff and University of Florida Staff
Highway Network	HNET	NET	Input	\Base\Input	2015 Road Characteristics Inventory (RCI), FDOT
	VFACTORS	CSV	Parameters	\Parameters	Gainesville 2010
	SPDCAP	DBF	Parameters	\Parameters	Gainesville 2010
	TURN	PEN	Input	\Base\Input	Gainesville 2010
Trip Distribution	FF	DBF	Parameters	\Parameters	Gainesville 2010
Transit Network	ALACHUAWLB.FAC	FAC	Parameters	\Parameters	Gainesville 2015
	ALACHUAWKPREM.FAC	FAC	Parameters	\Parameters	Gainesville 2015
	ALACHUA	FAR	Input	\Base\Input	Gainesville 2010
	SPDCRV	CSV	Parameters	\Parameters	Gainesville 2010
	TROUTE	LIN	Input	\Base\Input	Gainesville 2015
	ALACHUA	PTS	Parameters	\Parameters	Gainesville 2010

FDOT = Florida Department of Transportation
FTI = Florida Transportation Information
MTPO = Metropolitan Transportation Planning Organization
RCI = Roadway Characteristic Inventory
UF = University of Florida



Trip Generation



GNGEN00A.S

```
; Do not change filenames or add or remove FILEI/FILEO statements using an editor. Use
Cube/Application Manager.
RUN PGM=GENERATION PRNFILE="{SCENARIO_DIR}\output\GNGEN00A.PRN" MSG='Trip Generation'
FILEI LOOKUPI[3] = "{CATALOG_DIR}\parameters\ARATES.DBF"
FILEI LOOKUPI[2] = "{CATALOG_DIR}\parameters\DUWEIGHTS.DBF"
FILEO PRINTO[3] = "{SCENARIO_DIR}\output\AO_ERRORS.PRN"
FILEI LOOKUPI[1] = "{CATALOG_DIR}\parameters\GRATES.dbf"
FILEO PRINTO[2] = "{SCENARIO_DIR}\output\LUERRORS.PRN"
FILEO PRINTO[1] = "{SCENARIO_DIR}\output\GEN_SUM.PRN"
FILEO PAO[1] = "{SCENARIO_DIR}\output\PANDA_TEM.DBF",
 LIST=Z,P[1],P[2],P[3],P[4],P[5],P[6],P[7],P[8],P[9],P[10],P[11],P[12],
        A[1],A[2],A[3],A[4],A[5],A[6],A[7],A[8],A[9],A[10],A[11],A[12],DBF=T
FILEI ZDATI[3] = "{SCENARIO_DIR}\input\EITRIPS_{Year}{alt}.DBF"
FILEI ZDATI[2] = "{SCENARIO_DIR}\input\SPGEN_{Year}{alt}.DBF"
FILEI ZDATI[1] = "{SCENARIO_DIR}\input\ZoneData{YEAR}.dbf",
Z=TAZ_20{year}
; ================================================================
; OVERALL PROGRAM CONTROLS
PARAMETERS ZONES={ZONESA}, ZONEMSG=100
ARRAY CAR=4 CELL=999 CELLT=999 SPFRAC=12
LOOKUP LOOKUPI=1,
       NAME=PRATE, ;TRIP PRODUCTION RATES
         LOOKUP[1]=PAD, RESULT=RATEHBW,
         LOOKUP[2]=PAD, RESULT=RATEHBSH,
         LOOKUP[3]=PAD, RESULT=RATEHBSR,
         LOOKUP[4]=PAD, RESULT=RATEHBO,
       FAIL[1]=0,FAIL[2]=0,FAIL[3]=0, INTERPOLATE=N

LOOKUP LOOKUPI=2,
       NAME=DUWEIGHT,
         LOOKUP[1]=SIZERANGE, RESULT=PCT1PER, ; %1 PERSON
         LOOKUP[2]=SIZERANGE, RESULT=PCT2PER, ; %2 PERSON
         LOOKUP[3]=SIZERANGE, RESULT=PCT3PER, ; %3 PERSON
         LOOKUP[4]=SIZERANGE, RESULT=PCT4PER, ; %4 PERSON
         LOOKUP[5]=SIZERANGE, RESULT=PCT5PER, ; %5+PERSON
       INTERPOLATE=N, LIST=N

LOOKUP LOOKUPI=3,
       NAME=ARATE, ;TRIP ATTRACTION RATES
       LOOKUP[1]=PURPOSE, RESULT=ARATEOIE,
       LOOKUP[2]=PURPOSE, RESULT=ARATEMFG,
       LOOKUP[3]=PURPOSE, RESULT=ARATECOM,
       LOOKUP[4]=PURPOSE, RESULT=ARATESVC,
       LOOKUP[5]=PURPOSE, RESULT=ARATETOTE,
       LOOKUP[6]=PURPOSE, RESULT=ARATEDUS,
       LOOKUP[7]=PURPOSE, RESULT=ARATESCH,
     FAIL[1]=0,FAIL[2]=0,FAIL[3]=0, INTERPOLATE=N, LIST=N

PROCESS PHASE=ILOOP
; ================================================================
; This is the main program loop to calculate initial production
; and attraction values for each zone for each trip purpose
; ================================================================
LOOP HHTYPE=1,3 ; ESTABLISH VARIABLES FOR EACH HOUSEHOLD CLASS
IF (HHTYPE=1) ; SINGLE-FAMILY
   UNITS=   ZI.1.SFDU
   VACRATE= ZI.1.SF_SEA+ZI.1.SF_VAC
   PERMVACRATE= ZI.1.SF_VAC
   POP=     ZI.1.SPOP
   CAR[1]=ZI.1.SF_0V/100, CAR[2]=ZI.1.SF_1V/100, CAR[3]=ZI.1.SF_2V/100, CAR[4]=ZI.1.SF_3V/100
   IF(zi.1.MFPOP<0.8*zi.1.UF_OC_ST)
   POP=(ZI.1.SPOP+ZI.1.MFPOP-0.8*Zi.1.UF_OC_ST)
   UNITS=   ZI.1.SFDU+ZI.1.MFDU-0.4*Zi.1.UF_OC_ST
   ENDIF
ELSEIF (HHTYPE=2); MULTI-FAMILY
     UNITS=    ZI.1.MFDU ;assume 80% of the students making school related trips
     VACRATE= ZI.1.MF_SEA+ZI.1.MF_VAC
   PERMVACRATE= ZI.1.MF_VAC
     POP=     ZI.1.MFPOP;assume 80% of the students making school related trips
   IF(zi.1.MFPOP>0.8*zi.1.UF_OC_ST)
   POP=ZI.1.MFPOP-0.8*Zi.1.UF_OC_ST
```

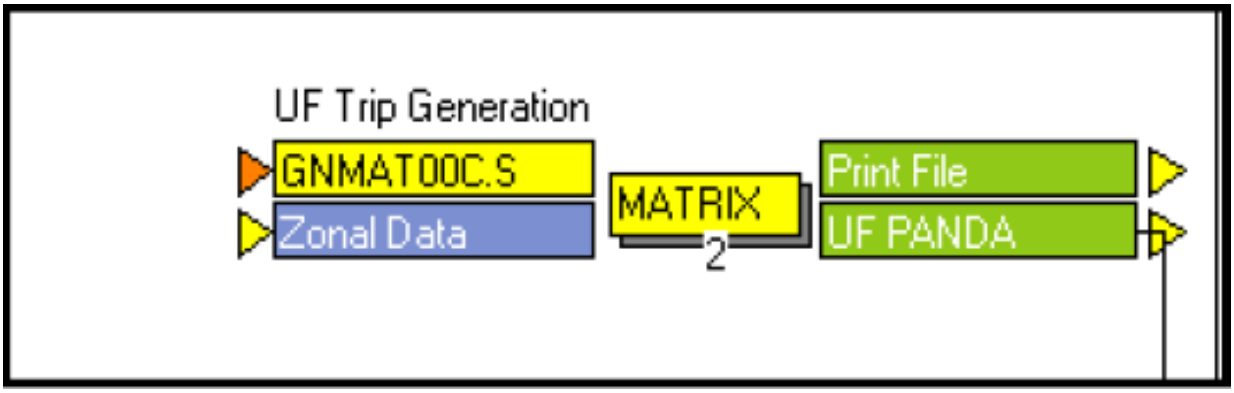
```
   UNITS=ZI.1.MFDU-0.4*Zi.1.UF_OC_ST
   ENDIF
   CAR[1]=ZI.1.MF_0V/100, CAR[2]=ZI.1.MF_1V/100, CAR[3]=ZI.1.MF_2V/100, CAR[4]=ZI.1.MF_3V/100
ELSEIF (HHTYPE=3) ; HOTEL/MOTEL
   UNITS=   ZI.1.HMDU
   VACRATE= 100-ZI.1.HM_POC
   PERMVACRATE= 100-ZI.1.HM_POC
   POP=     ZI.1.HMPOP
   CAR[1]=0, CAR[2]=1.0, CAR[3]=0, CAR[4]=0
ENDIF
; ===================================================================
; From here down, the same equations get applied to each
; household size, auto ownership and dwelling unit type.
; Since it is being run in the HHTYPE loop, the same equations
; will be applied and running totals by zone will be accumulated.
; ===================================================================
VAC=UNITS*(VACRATE/100), OCC=UNITS-VAC, GENVAC=UNITS*(PERMVACRATE/100), GENOCC=UNITS-GENVAC
IF (HHTYPE<>3) TOCC=TOCC+OCC ;Keep track of total permenantly occupied DUs
TGOCC=TGOCC+GENOCC ;Keep track of total occupied DUs
IF (OCC>0)
   POPDU=POP/OCC
ELSE
   POPDU=0
ENDIF
IF (POPDU<>0) PDUCNT=PDUCNT+1      ;Keep track of total zones with pop/du ratios
IF (POPDU<>0) PDUTOT=PDUTOT+POPDU  ;total pop/du ratios
IF (POPDU<=1.12) RANGE=1
IF (POPDU>1.12)  RANGE=2
IF (POPDU>1.37)  RANGE=3
IF (POPDU>1.62)  RANGE=4
IF (POPDU>1.87)  RANGE=5
IF (POPDU>2.12)  RANGE=6
IF (POPDU>2.37)  RANGE=7
IF (POPDU>2.62)  RANGE=8
IF (POPDU>2.87)  RANGE=9
IF (POPDU>3.12)  RANGE=10
IF (POPDU>3.37)  RANGE=11
IF (POPDU>3.62)  RANGE=12
IF (POPDU>3.87)  RANGE=13
IF (POPDU>4.12)  RANGE=14
IF (POPDU>4.37)  RANGE=15
IF (POPDU>4.62)  RANGE=16
IF (POPDU>5.99)  RANGE=17
IF  (POPDU<1&OCC>0)  PRINT  LIST="POP/DU  ERROR,  HHTYPE=",HHTYPE(1.0),"  Population=",POP(4.0C),"
Occupied Units=",occ(4.0c)," TAZ=",TAZ_2015(4.0c) printo=2
  LOOP PR=1,5
   LOOP AU=1,4
         CL=100*PR+10*(AU-1)+HHTYPE
         CELL[CL]=GENOCC*DUWEIGHT(PR,RANGE)*CAR[AU]
         CELLT[CL]=CELLT[CL]+CELL[CL]
         LOOP PURP=1,4
            PRODRATE=PRATE(PURP,CL)
            P[PURP]=P[PURP]+PRATE(PURP,CL)*CELL[CL]
         ENDLOOP
   ENDLOOP
  ENDLOOP
ENDLOOP ; ON HHTYPE
;====================================================================
;pre-process prior to attractions calculation
;minor employment adjustment using UF_EMP Data
;(1) Subtract UF employment from service employment if UF<Service
 COMEMP=ZI.1.COMEMP
 SERVEMPx=ZI.1.SERVEMP
IF(ZI.1.SERVEMP>ZI.1.UF_EMP)
 SERVEMPx=ZI.1.SERVEMP-ZI.1.UF_EMP
 IF(ZI.1.UF_EMP>0)
  PRINT LIST='EMPLOYMENT ZONE', I
 ENDIF
 IF(ZI.1.UF_PARKING>0)
  PRINT LIST='PARKING    ZONE', I
 ENDIF
```

```
ELSE
;(2) Take the remaining UF service employment from commercial if UF>Service
 COMEMP=ZI.1.COMEMP-ZI.1.UF_EMP+ZI.1.SERVEMP
 SERVEMPx =  0
  IF (COMEMP<0)
  COMEMP=0
  ENDIF
ENDIF
;(3) Compute total UF parking and employment
; replaced hard-coded zone number below
UF_EMP1=0
UF_PRK1=0
LOOP II=1,{ZONESA}
 UF_EMP1=UF_EMP1+ZI.1.UF_EMP[II]
 UF_PRK1=UF_PRK1+ZI.1.UF_PARKING[II]
ENDLOOP
PRINT LIST=UF_EMP1, '   ', UF_PRK1
;(4) Allocate UF employment to parking TAZs based on proportion of parking spaces
 ;SERVEMP= ZI.1.SERVEMP[I] + UF_EMP1*(ZI.1.UF_PARKING[I]/UF_PRK1)
  SERVEMP= SERVEMPx + UF_EMP1*(ZI.1.UF_PARKING/UF_PRK1)
 TOTALEMP=ZI.1.MFGEMP+ZI.1.OIEMP+COMEMP+SERVEMP
;=====================================================================
; Now process the trip purposes that are attraction-based
; =====================================================================
; PURPOSE 1 = HBW
; PURPOSE 2 = HBSH
; PURPOSE 3 = HBSR
; PURPOSE 4 = HBO
; PURPOSE 5 = NHB
; PURPOSE 6 = 4 Tire Truck
; PURPOSE 7 = Single-Unit Truck
; PURPOSE 8 = Tractor-trailer
TOTALDUS=ZI.1.SFDU+ZI.1.MFDU
LOOP WPURP=1,8
   A[WPURP]=ARATE(1,WPURP)*ZI.1.MFGEMP+
            ARATE(2,WPURP)*ZI.1.OIEMP+
            ARATE(3,WPURP)*COMEMP+
            ARATE(4,WPURP)*SERVEMP+
            ARATE(5,WPURP)*TOTALEMP+
            ARATE(6,WPURP)*TOTALDUS+
            ARATE(7,WPURP)*ZI.1.SCHENR
ENDLOOP
P[5]=A[5]
P[6]=A[6]
P[7]=A[7]
P[8]=A[8]
; ....................................................................
; For the next four purposes,
; Attractions are a function of the total attractions to a zone.
; Since totals aren't known until we finish the initial calculations,
; attractions for these purposes will be initially calculated in the
; ADJUST PHASE.
; ....................................................................
; PURPOSE 9 = SOV EI
; PURPOSE 10 = HOV EI
; PURPOSE 11 = LDTK EI
; PURPOSE 12 = HDTK EI
; SOV EI
P[9]=ZI.3.TRIPS*(ZI.3.LOVPCT/100)
; HOV EI
P[10]=ZI.3.TRIPS*(ZI.3.HOVPCT/100)
; LDTK EI
P[11]=ZI.3.TRIPS*(ZI.3.LDTPCT/100)
; HDTK EI
P[12]=ZI.3.TRIPS*(ZI.3.HDTPCT/100)
; =====================================================================
; NOW PROCESS SPECIAL GENERATORS
SPFRAC[1]=ZI.2.HBWP/100
SPFRAC[2]=ZI.2.HBSHP/100
SPFRAC[3]=ZI.2.HBSRP/100
SPFRAC[4]=ZI.2.HBOP/100
```

```
SPFRAC[5]=ZI.2.NHBP/100
SPFRAC[6]=ZI.2.TRK4P/100
SPFRAC[7]=ZI.2.TRKSUNITP/100
SPFRAC[8]=ZI.2.TRKCOMBOP/100
SPFRAC[9]=ZI.2.EILOVP/100
SPFRAC[10]=ZI.2.EIHOVP/100
SPFRAC[11]=ZI.2.EILDTP/100
SPFRAC[12]=ZI.2.EIHDTP/100

LOOP PRP=1,12
    IF (ZI.2.PROD='Y','y')
       IF (ZI.2.FUNCTIONP='+') P[PRP]=P[PRP]+VALUEP*SPFRAC[PRP]
       IF (ZI.2.FUNCTIONP='-') P[PRP]=P[PRP]-VALUEP*SPFRAC[PRP]
    ENDIF
ENDLOOP
SPFRAC[1]=ZI.2.HBWA/100
SPFRAC[2]=ZI.2.HBSHA/100
SPFRAC[3]=ZI.2.HBSRA/100
SPFRAC[4]=ZI.2.HBOA/100
SPFRAC[5]=ZI.2.NHBA/100
SPFRAC[6]=ZI.2.TRK4A/100
SPFRAC[7]=ZI.2.TRKSUNITA/100
SPFRAC[8]=ZI.2.TRKCOMBOA/100
SPFRAC[9]=ZI.2.EILOVA/100
SPFRAC[10]=ZI.2.EIHOVA/100
SPFRAC[11]=ZI.2.EILDTA/100
SPFRAC[12]=ZI.2.EIHDTA/100
LOOP PRP=1,12
    IF (ZI.2.ATTR='Y','y')
       IF (ZI.2.FUNCTIONA='+') A[PRP]=A[PRP]+VALUEA*SPFRAC[PRP]
       IF (ZI.2.FUNCTIONA='-') A[PRP]=A[PRP]-VALUEA*SPFRAC[PRP]
    ENDIF
ENDLOOP

;*************************************************************************
;This portion of the script checks to see if any zones with populations are lacking values for
percent automobile ownership.  If so, the model crashes and reports the problem zones so that the
user can correct the problem.  All zones with populations should have values for percent automobile
ownership or the model will not generate Home-Based trips for those zones.
IF (I=1)
  PRINT LIST='\nAUTO OWNERSHIP ERRORS WHERE POPULATION EXISTS BUT AUTO OWNERSHIP DOES NOT', PRINTO=3
  PRINT LIST='\nCHECK LISTED ZONES IN ZONEDATA{Year} FILES FOR AUTO OWNERSHIP PERCENTAGES!!!',
PRINTO=3
  PRINT LIST='\n', PRINTO=3
  SFAOERROR=0
  MFAOERROR=0
ENDIF
     SFAO=zi.1.SF_0V+zi.1.SF_1V+zi.1.SF_2V+zi.1.SF_3V
     MFAO=zi.1.MF_0V+zi.1.MF_1V+zi.1.MF_2V+zi.1.MF_3V
     IF ((zi.1.SPOP<>0 & SFAO=0)|(zi.1.MFPOP<>0 & MFAO=0))
        PRINT LIST='\n', PRINTO=3
     ENDIF
     IF (zi.1.SPOP<>0 & SFAO=0)
        SFAOERROR=SFAOERROR+1
        PRINT LIST='\nAUTO OWNERSHIP = 0 BUT SF POPULATION > 0 ERROR FOR ZONE=',I(5.0),PRINTO=3
     ELSE
     ENDIF
     IF (zi.1.MFPOP<>0 & MFAO=0)
        MFAOERROR=MFAOERROR+1
        PRINT LIST='\nAUTO OWNERSHIP = 0 BUT MF POPULATION > 0 ERROR FOR ZONE=',I(5.0),PRINTO=3
     ELSE
     ENDIF

IF (I={ZONESA})
  PRINT LIST='\n************Error Report Summary*************',
             '\nTOTAL AUTO OWNERSHIP ERRORS FOR SINGLE FAMILY=',SFAOERROR(8.0C),
             '\nTOTAL AUTO OWNERSHIP ERRORS FOR MULTI FAMILY= ',MFAOERROR(8.0C), printo=3
SFAOERROR=0
MFAOERROR=0 ;set both to 0 in order to sucess executive by Mia
IF (SFAOERROR=0 & MFAOERROR=0)  PRINT LIST='\n',
```

```
                                  '\nTHERE ARE NO AUTO OWNERSHIP = 0 BUT POPULATION > 0 ERRORS',
PRINTO=3
  if (SFAOERROR>1) abort
  if (MFAOERROR>1) abort

ENDIF

;*********************************************************************
; ===================================================================
ENDPROCESS

PROCESS PHASE=ADJUST
LOOP PURP=1,12
 IF (PURP=1) PRINT LIST="TRIP PRODUCTION AND ATTRACTION REPORT BY PURPOSE", PRINTO=1
 PRINT    LIST="       Purpose=",PURP(2.0),"       Productions=",P[PURP][0](12.0C),"    Unbalanced
Attractions=",A[PURP][0](12.0C), PRINTO=1
ENDLOOP
 PRINT LIST=" ", PRINTO=1
; . . . . . .
. . . . . .
; Balancing attractions as similarly done in Olympus model.
BALANCE A2P=1-4
; . . . . . .
. . . . . .
 TOTSTDATTR=A[1][0]+A[2][0]+A[3][0]+A[4][0]+A[5][0]
A[9]=P[9][0]*(A[1]+A[2]+A[3]+A[4]+A[5])/TOTSTDATTR
A[10]=P[10][0]*(A[1]+A[2]+A[3]+A[4]+A[5])/TOTSTDATTR
A[11]=P[11][0]*(A[7]/A[7][0])
A[12]=P[12][0]*(A[8]/A[8][0])
BALANCE A2P=9-12
LOOP PURP=1,12
 PRINT    LIST="      Purpose=",PURP(2.0),"      Productions=",P[PURP][0](12.0C),"         Balanced
Attractions=",A[PURP][0](12.0C), PRINTO=1
ENDLOOP

PTOTAL=P[1][0]+P[2][0]+P[3][0]+P[4][0]+P[5][0]+P[6][0]+P[7][0]+P[8][0]+P[9][0]+P[10][0]+P[11][0]+
P[12][0]
ATOTAL=A[1][0]+A[2][0]+A[3][0]+A[4][0]+A[5][0]+A[6][0]+A[7][0]+A[8][0]+A[9][0]+A[10][0]+A[11][0]+
A[12][0]

PRINT LIST="  Total","       Productions=",PTOTAL(12.0C),"            Attractions=",ATOTAL(12.0C),
PRINTO=1
PRINT LIST=" ", PRINTO=1
  POPTOT=ZI.1.SPOP[0]+ZI.1.MFPOP[0]
  ALLPOP=POPTOT+ZI.1.HMPOP[0]
  PDUAVG=(POPTOT/TOCC)
  ALPDAG=(ALLPOP/TGOCC)
  TOTSRV=ZI.1.SERVEMP[0]
  TOTCOM=ZI.1.COMEMP[0]
  TOTMFG=ZI.1.MFGEMP[0]
  TOTIND=ZI.1.OIEMP[0]
  TOTEMP=ZI.1.TOTEMP[0]
  EMPPOP=TOTEMP/POPTOT
  SRVRTE=TOTSRV/TOTEMP
  COMRTE=TOTCOM/TOTEMP
  MFGRTE=TOTMFG/TOTEMP
  INDRTE=TOTIND/TOTEMP
  IITRIP=P[1][0]+P[2][0]+P[3][0]+P[4][0]+P[5][0]+P[6][0]+P[7][0]+P[8][0]
  ITPPRM=IITRIP/TOCC
  ITPTOC=IITRIP/TGOCC
  ITPEMP=IITRIP/TOTEMP
  ITPPOP= IITRIP/POPTOT
 PRINT LIST="  Permanent Population =                                    ",POPTOT(12.0C), PRINTO=1
PRINT LIST="  Total Population =                                       ",ALLPOP(12.0C), PRINTO=1
PRINT LIST="  Permanently Occupied Dwelling Units =                    ",TOCC(12.0C), PRINTO=1
PRINT LIST="  Transient and Permently Occupied Dwelling Units =        ",TGOCC(12.0C), PRINTO=1
PRINT LIST="  Total Service Employment =                               ",TOTSRV(12.0C), PRINTO=1
PRINT LIST="  Total Commercial Employment =                            ",TOTCOM(12.0C), PRINTO=1
PRINT LIST="  Total Manufacturing Employment =                         ",TOTMFG(12.0C), PRINTO=1
PRINT LIST="  Total Other Industrial Employment =                      ",TOTIND(12.0C), PRINTO=1
PRINT LIST="  Total Employment =                                       ",TOTEMP(12.0C), PRINTO=1
```

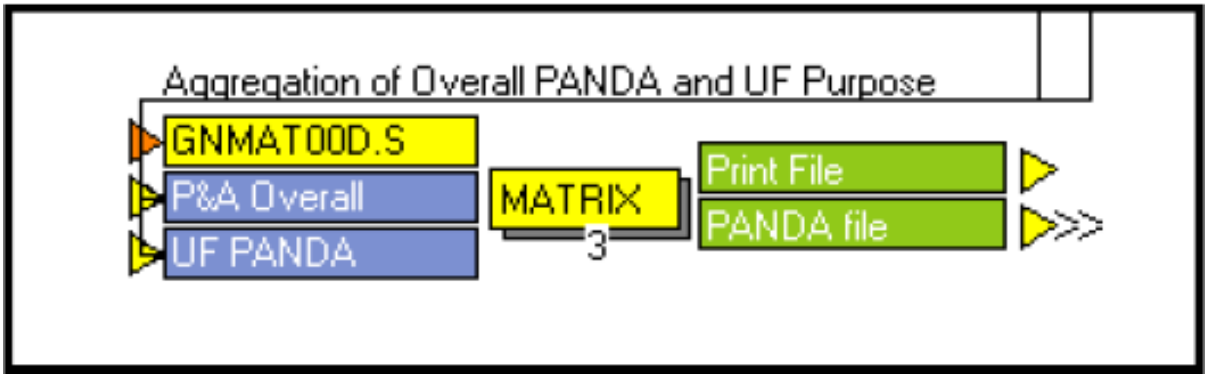
```
PRINT LIST="   Permanent Population per Permenantly Occupied Dwelling Unit =   ",PDUAVG(5.2C),
PRINTO=1
PRINT LIST="  Total Population per Total Occupied Dwelling Unit =              ",ALPDAG(5.3C),
PRINTO=1
PRINT LIST="  Total Employment per Permanent Population =                      ",EMPPOP(5.3C),
PRINTO=1
PRINT LIST="  Service to Total Employment =                                    ",SRVRTE(5.3C),
PRINTO=1
PRINT LIST="  Commercial to Total Employment =                                 ",COMRTE(5.3C),
PRINTO=1
PRINT LIST="  Manufacturing to Total Employment =                              ",MFGRTE(5.3C),
PRINTO=1
PRINT LIST="  Other Industrial to Total Employment =                           ",INDRTE(5.3C),
PRINTO=1
PRINT LIST="   Internal Person Trips per Permanently Occupied Dwelling Unit =  ",ITPPRM(5.3C),
PRINTO=1
PRINT LIST="   Internal Person Trips per Total Occupied Dwelling Units =       ",ITPTOC(5.3C),
PRINTO=1
PRINT LIST="  Internal Person Trips per Employee =                             ",ITPEMP(5.3C),
PRINTO=1
PRINT LIST="  Internal Person Trips per Person =                          ",ITPPOP(5.3C), PRINTO=1
ENDPROCESS
ENDRUN
```



GNMAT00C.S

```
; Do not change filenames or add or remove FILEI/FILEO statements using an editor. Use
Cube/Application Manager.
RUN PGM=MATRIX PRNFILE="{SCENARIO_DIR}\output\UFGEN.prn" MSG='UF Trip Generation'
FILEI ZDATI[1] = "{SCENARIO_DIR}\input\ZoneData{YEAR}.DBF",
Z=TAZ_20{year}
FILEO RECO[1] = "{SCENARIO_DIR}\output\UFPANDA.dbf",
                FIELDS=Z,HBUP,HBUA,HDORMUP,HDORMUA,STUPCT,nocarpct,wcarpct
;FILEO PRINTO[1] = "{SCENARIO_DIR}\output\GEN_SUM.PRN"
PAR ZONES={ZONESA}
; Trip rates from HH survey
; Off-campus student trips
RO.HBUP = {RATE_HBUP}*ZI.1.UF_OC_ST*{HBO-TF}       ; home-based university PRODS from off-campus
(students)
;RO.HBUA = {RATE_HBUA}*ZI.1.UF_PARKING*{HBO-TF}     ; home-based university ATTRS from off-campus
(parking spaces)
RO.HBUA = {RATE_HBUA}*ZI.1.UF_ST_PARK*{HBO-TF}   ;Use Separated students parking than the whole UF
parking.
; Campus housing student trips
RO.HDORMUP = {RATE_HDORMUP} *ZI.1.UF_DORM_ST*{HBO-TF} ; home-based university PRODS from Campus
housing (students)
RO.HDORMUA = {RATE_HDORMUA} *ZI.1.SEATS*{HBO-TF}     ; home-based university ATTRS from classroom
seats
ufpop = ZI.1.UF_OC_ST + ZI.1.UF_DORM_ST ; UF pop is equal to number of off-campus students plus
dorm students
sfpop = ZI.1.SPOP
mfpop = ZI.1.MFPOP
tpop=sfpop+mfpop
```

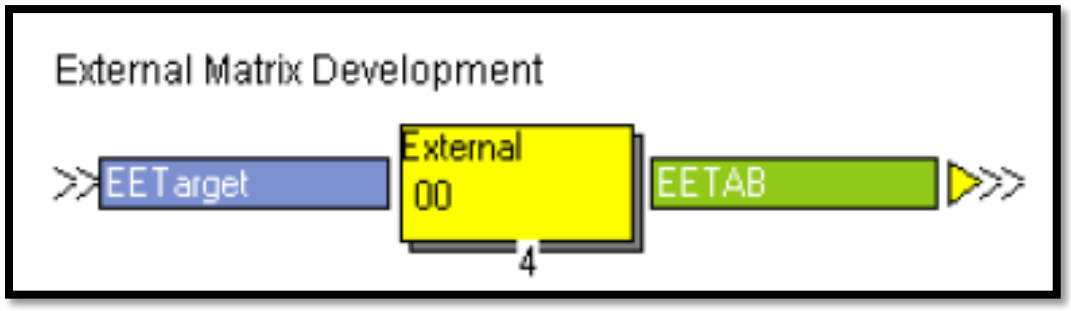
```
sf0 = 0.01*ZI.1.SF_0V
mf0 = 0.01*ZI.1.MF_0V
;Student market share
if (tpop>0)
  RO.STUPCT=ufpop/tpop
  t0=(sf0*sfpop + mf0*mfpop)/tpop
else
  RO.STUPCT=0.0
  t0=0.0
endif
if (STUPCT>1.0) STUPCT=1.0    ; make sure fraction students not greater than 1.0
nocarpct= t0*(1.0-STUPCT)     ; fraction without autos
wcarpct = 1.0-nocarpct-STUPCT ; fraction with autos
WRITE RECO=1
ENDRUN
```

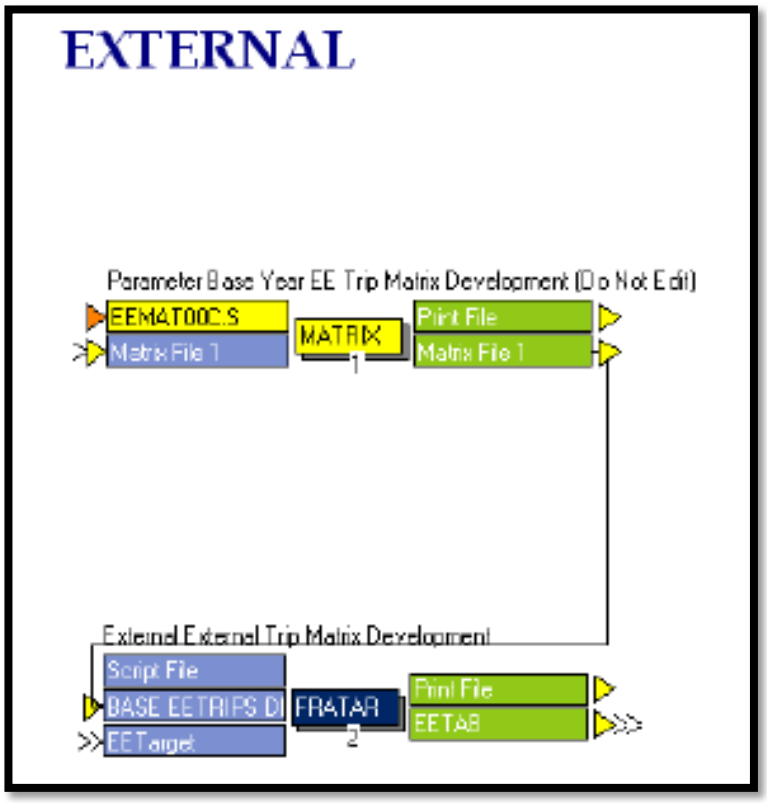


GNMAT00D.S

```
; Do not change filenames or add or remove FILEI/FILEO statements using an editor. Use
Cube/Application Manager.
RUN PGM=MATRIX PRNFILE="{SCENARIO_DIR}\output\GNMAT00B.PRN" MSG='Aggregation of Overall PANDA and
UF Purpose'
FILEI ZDATI[2] = "{SCENARIO_DIR}\output\UFPANDA.dbf"
FILEI ZDATI[1] = "{SCENARIO_DIR}\output\PANDA_TEM.DBF"
FILEO RECO[1] = "{SCENARIO_DIR}\output\PANDA.DBF",
FIELDS=Z,HBWP,HBWA,HBSHP,HBSHA,HBSRP,HBSRA,HBOP,HBOA,NHBP,NHBA,                                TK4P,
TK4A,SGLUNITP,SGLUNITA,TRKTRLRP,TRKTRLRA,SOVIEP,SOVIEA,HOVIEP,
HOVIEA,LDTKIEP,LDTKIEA,HDTKIEP,HDTKIEA,HBUP,HBUA,HDORMUP,HDORMUA

PAR ZONES={ZONESA}

RO.HBWP=ZI.1.P1
RO.HBSHP=ZI.1.P2
RO.HBSRP=ZI.1.P3
RO.HBOP=ZI.1.P4
RO.NHBP=ZI.1.P5
RO.TK4P=ZI.1.P6
RO.SGLUNITP=ZI.1.P7
RO.TRKTRLRP=ZI.1.P8
RO.SOVIEP=ZI.1.P9
RO.HOVIEP=ZI.1.P10
RO.LDTKIEP=ZI.1.P11
RO.HDTKIEP=ZI.1.P12
RO.HBWA=ZI.1.A1
RO.HBSHA=ZI.1.A2
RO.HBSRA=ZI.1.A3
RO.HBOA=ZI.1.A4
RO.NHBA=ZI.1.A5
RO.TK4A=ZI.1.A6
RO.SGLUNITA=ZI.1.A7
RO.TRKTRLRA=ZI.1.A8
RO.SOVIEA=ZI.1.A9
RO.HOVIEA=ZI.1.A10
RO.LDTKIEA=ZI.1.A11
```

```
RO.HDTKIEA=ZI.1.A12
RO.HBUP=ZI.2.hbup
RO.HBUA=ZI.2.hbua
RO.HDORMUP=ZI.2.hdormup
RO.HDORMUA=ZI.2.hdormua
WRITE RECO=1

ENDRUN
```

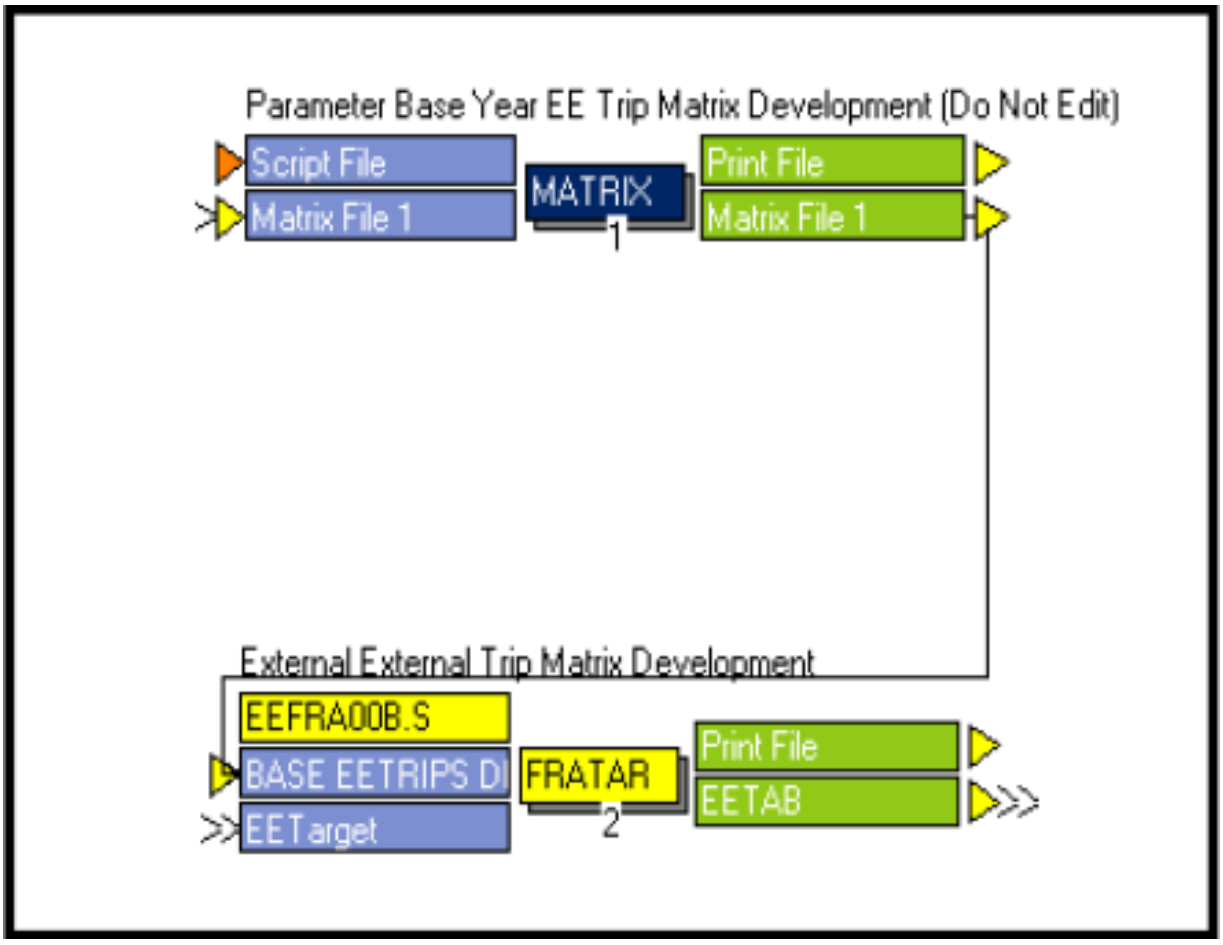




EEMAT00C.S

```
; Do not change filenames or add or remove FILEI/FILEO statements using an editor. Use
Cube/Application Manager.
RUN PGM=MATRIX PRNFILE="{SCENARIO_DIR}\output\EEMAT00c.PRN" MSG='Parameter Base Year EE Trip Matrix
Development (Do Not Edit)'
FILEO MATO[1] = "{CATALOG_DIR}\PARAMETERS\BASEYEAR_EETRIPS_DIST.MAT",
MO=1
FILEI MATI[1] = "{CATALOG_DIR}\PARAMETERS\BASEYEAR_EETRIPS_IJ.DBF",
 pattern=ijm:v, fields=orz,dsz,0,autotrips
PAR zones={ZONESA}
```

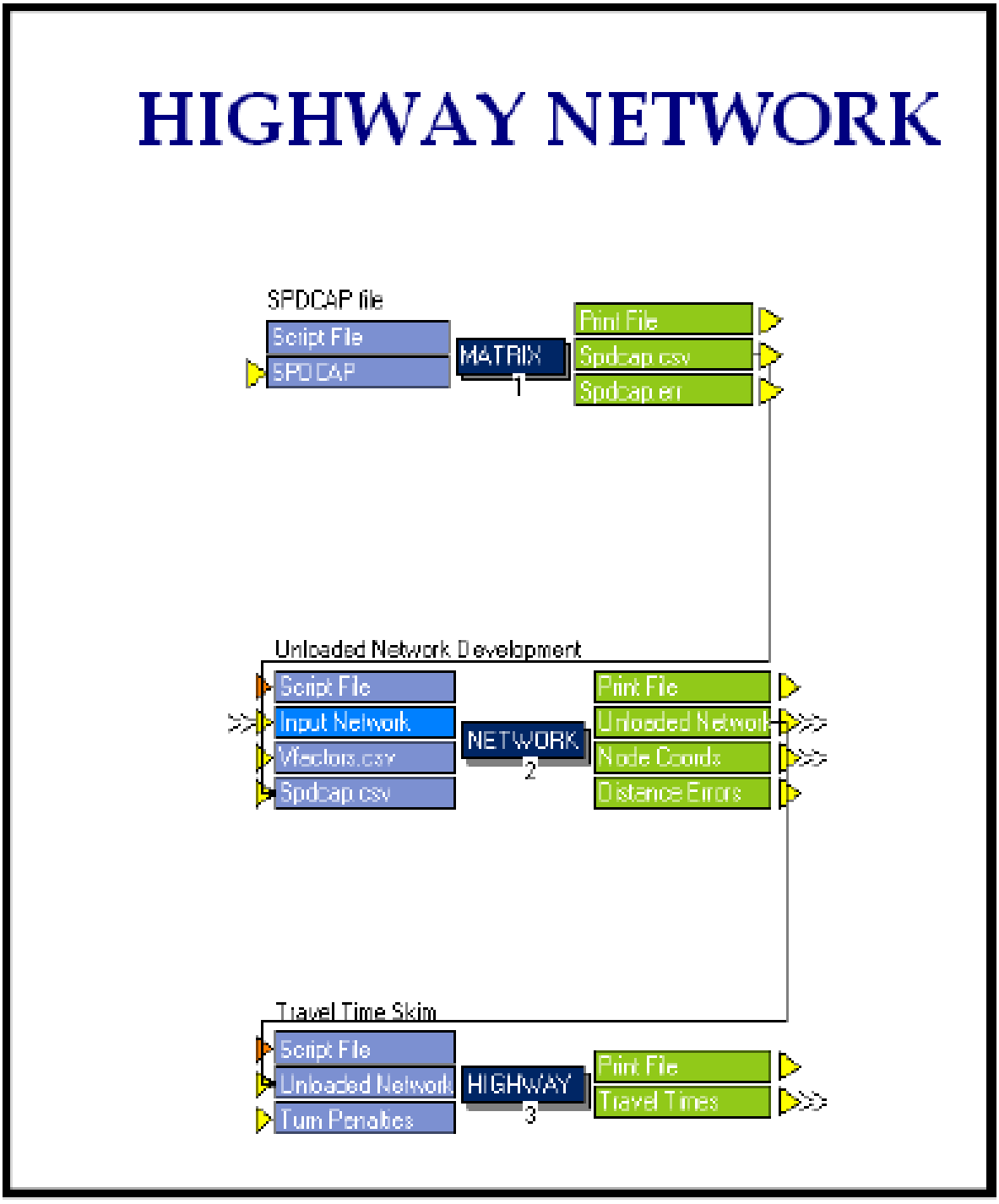
```
mw[1]=mi.1.1
ENDRUN
```

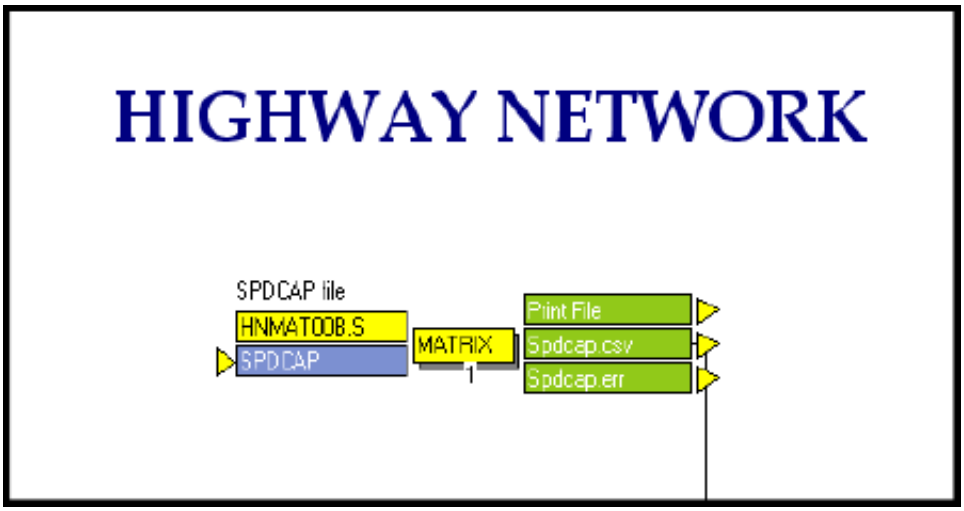


EEFRA00B.S

```
; Do not change filenames or add or remove FILEI/FILEO statements using an editor. Use
Cube/Application Manager.
RUN PGM=FRATAR PRNFILE="{SCENARIO_DIR}\output\EEFRA00A.PRN" MSG='External External Trip Matrix
Development'
FILEI MATI[1] = "{CATALOG_DIR}\PARAMETERS\BASEYEAR_EETRIPS_DIST.MAT"
FILEO MATO[1] = "{SCENARIO_DIR}\output\EETAB.MAT",
MO=1-2, name=EETRIPS,EETTRIPS
FILEI ZDATI[1] = "{SCENARIO_DIR}\INPUT\eeTARGET20{YEAR}.dbf"
MAXITERS=99
SETPA P[1]=ZI.1.EEO, A[1]=ZI.1.EED MW[1]=MI.1.1
SETPA P[2]=ZI.1.EETO, A[2]=ZI.1.EETD MW[2]=MI.1.1
ACOMP=1,PCOMP=1
MARGINS=1

ENDRUN
```

Highway Network Step

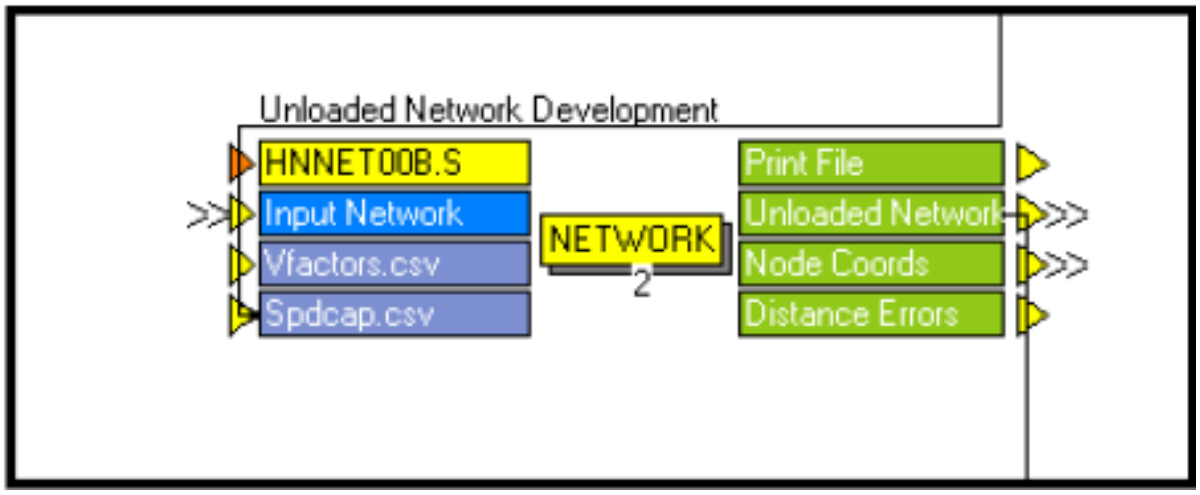




HNMAT00B.S

```
; Do not change filenames or add or remove FILEI/FILEO statements using an editor. Use
Cube/Application Manager.
RUN PGM=MATRIX PRNFILE="{SCENARIO_DIR}\Output\SPDCAP.OUT" MSG='SPDCAP file'
FILEO PRINTO[2] = "{SCENARIO_DIR}\OUTPUT\SPDCAP.ERR"
FILEO PRINTO[1] = "{SCENARIO_DIR}\output\SPDCAP.CSV"
FILEI RECI = "{CATALOG_DIR}\PARAMETERS\SPDCAP.DBF"

ARRAY SPDLOOKUP=999999 CAPLOOKUP=999999
    _LATVAL=RI.LOW_ATYPE
    _HATVAL=RI.HIGH_ATYPE
    _LFTVAL=RI.LOW_FTYPE
    _HFTVAL=RI.HIGH_FTYPE
    _LLNVAL=RI.LOW_LANES
    _HLNVAL=RI.HIGH_LANES
    _CAPVAL=RI.CAPACITY
    _SPDVAL=RI.SPEED
    _CAPFUNC=RI.CAP_OPERAN
    _SPDFUNC=RI.SPEED_OPER
; PLACE INITIAL CAPACITIES & SPEEDS INTO AN ARRAY
IF (_CAPFUNC=' ')
    LOOP ATYPE=_LATVAL,_HATVAL
      LOOP FTYPE=_LFTVAL,_HFTVAL
        LOOP LANES=_LLNVAL,_HLNVAL
        INDEXVAL=ATYPE*10000+FTYPE*100+LANES
        CAPLOOKUP[INDEXVAL]=_CAPVAL
        ENDLOOP
      ENDLOOP
    ENDLOOP
ENDIF
IF (_SPDFUNC=' ')
    LOOP ATYPE=_LATVAL,_HATVAL
      LOOP FTYPE=_LFTVAL,_HFTVAL
        LOOP LANES=_LLNVAL,_HLNVAL
        INDEXVAL=ATYPE*10000+FTYPE*100+LANES
        SPDLOOKUP[INDEXVAL]=_SPDVAL
        ENDLOOP
      ENDLOOP
    ENDLOOP
ENDIF
IF (_CAPFUNC='*')
    LOOP ATYPE=_LATVAL,_HATVAL
      LOOP FTYPE=_LFTVAL,_HFTVAL
        LOOP LANES=_LLNVAL,_HLNVAL
        INDEXVAL=ATYPE*10000+FTYPE*100+LANES
        CAPLOOKUP[INDEXVAL]=CAPLOOKUP[INDEXVAL]*_CAPVAL
        ENDLOOP
      ENDLOOP
    ENDLOOP
```

```
ENDIF
IF (_SPDFUNC='*'|_SPDFUNC='+'|_SPDFUNC='-')
    LOOP ATYPE=_LATVAL,_HATVAL
      LOOP FTYPE=_LFTVAL,_HFTVAL
        LOOP LANES=_LLNVAL,_HLNVAL
        INDEXVAL=ATYPE*10000+FTYPE*100+LANES
        IF (_SPDFUNC='*') SPDLOOKUP[INDEXVAL]=SPDLOOKUP[INDEXVAL]*_SPDVAL
        IF (_SPDFUNC='+') SPDLOOKUP[INDEXVAL]=SPDLOOKUP[INDEXVAL]+_SPDVAL
        IF (_SPDFUNC='-') SPDLOOKUP[INDEXVAL]=SPDLOOKUP[INDEXVAL]-_SPDVAL
        ENDLOOP
      ENDLOOP
    ENDLOOP
ENDIF
IF (I=0)
  PRINT LIST='SPEED OR CAPACITY ERRORS WHERE THE SPDCAP RESULT IS LESS THAN ZERO', PRINTO=2
  LOOP IVAL=1,999999
     IF         (CAPLOOKUP[IVAL]>0|SPDLOOKUP[IVAL]>0)                          PRINT         CSV=T,
LIST=IVAL(6.0),CAPLOOKUP[IVAL],SPDLOOKUP[IVAL],PRINTO=1
     IF (CAPLOOKUP[IVAL]<0)
        CAPERRCNT=CAPERRCNT+1
        PRINT        CSV=T,        LIST='SPDCAP        ERROR        FOR        ATFTLN=',IVAL(6.0),'
CAPACITY=',CAPLOOKUP[IVAL](9.2),PRINTO=2
     ENDIF
     IF (SPDLOOKUP[IVAL]<0)
        SPDERRCNT=SPDERRCNT+1
        PRINT        CSV=T,        LIST='SPDCAP        ERROR        FOR        ATFTLN=',IVAL(6.0),'
SPEED=',SPDLOOKUP[IVAL](9.2),PRINTO=2
     ENDIF
  ENDLOOP
  PRINT LIST='\n************Error Report Summary*************',
             '\nTOTAL LESS THAN ZERO CAPACITY ERRORS=',CAPERRCNT(8.0C),
             '\nTOTAL LESS THAN ZERO SPEED ERRORS   =',SPDERRCNT(8.0C), printo=2
ENDIF
ENDRUN
```

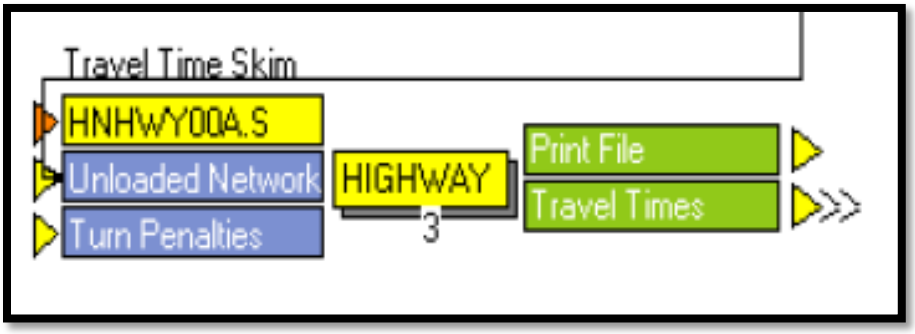


HNNET00B.S

```
; Do not change filenames or add or remove FILEI/FILEO statements using an editor. Use
Cube/Application Manager.
RUN PGM=NETWORK PRNFILE="{SCENARIO_DIR}\Output\HNNET00A.PRN" MSG='Unloaded Network Development'
FILEI LOOKUPI[1] = "{CATALOG_DIR}\parameters\VFACTORS.CSV"
FILEO PRINTO[2] = "{SCENARIO_DIR}\OUTPUT\NETERRORS.PRN"
FILEI LINKI[1] = "{SCENARIO_DIR}\INPUT\HNET20{YEAR}.NET"
FILEI LOOKUPI[2] = "{SCENARIO_DIR}\output\SPDCAP.CSV"
FILEO PRINTO[1] = "{SCENARIO_DIR}\OUTPUT\NODECOOR.CSV"
FILEO NETO = "{SCENARIO_DIR}\OUTPUT\UNLOADED.NET",  EXCLUDE=LINKCNT

PAR  LIST_ERRS=0 MAX_IP_ERRS=10000
ARRAY _ATCNT=99,_FTCNT=99
LOOKUP, NAME=VFACTORS,
    LOOKUP[1]=1, RESULT=2,
```

```
    LOOKUP[2]=1, RESULT=3,
    LOOKUP[3]=1, RESULT=4,
    LOOKUP[4]=1, RESULT=5,
    INTERPOLATE=N, LOOKUPI=1
LOOKUP, NAME=SPDCAP,
    LOOKUP[1]=1, RESULT=2,
    LOOKUP[2]=1, RESULT=3,
    INTERPOLATE=N, LOOKUPI=2

PROCESS  PHASE=INPUT
;Use this phase to modify data as it is read, such as recoding node numbers.
ENDPROCESS

PROCESS  PHASE=NODEMERGE
   print csv=t list=N(6.0),X,Y, PRINTO=1
ENDPROCESS

PROCESS  PHASE=LINKMERGE
COMP FTYPE=LI.1.FTYPE
COMP FTYPE1=INT(LI.1.FTYPE/10)
COMP ATYPE=LI.1.ATYPE
COMP ATYPE1=INT(LI.1.ATYPE/10)
COMP LANES=LI.1.LANES
IF (DISTANCE<=0)
  DISTANCE=SQRT((A.X-B.X)^2+(A.Y-B.Y)^2)/{UNITS}
endif

 _MYDIST=SQRT((A.X-B.X)^2+(A.Y-B.Y)^2)/{UNITS}
 _err=(_MYDIST-DISTANCE)/DISTANCE
if(_err >0.01) print list=A,B,_MYDIST(8.4),DISTANCE(8.4) PRINTO=2

 ; PUT VFACTORS ON NETWORK
  linkcnt=1
  UROADFACTOR=VFACTORS(1,FTYPE)
  CONFAC=VFACTORS(2,FTYPE)
  BPRCOEFFICIENT=VFACTORS(3,FTYPE)
  BPREXPONENT=VFACTORS(4,FTYPE)
; PUT SPEEDS AND CAPACITIES ON NETWORK
  _INDEXVAL=10000*ATYPE+100*FTYPE+Lanes
  CAPACITY=SPDCAP(1,_INDEXVAL)*Lanes

  IF(RESTRICTED=1) CAPACITY=round(CAPACITY*0.5)

  IF (CAPACITY=0)
     DAILYCAP=999999
  ELSE
      DAILYCAP=(CAPACITY/CONFAC)*UROADFACTOR
  ENDIF

  SPEED=SPDCAP(2,_INDEXVAL)
  IF (SPEED!=0)
    TIME=60*DISTANCE/SPEED
  ENDIF
  if (time<0.01) time=0.01
; PUT WALKTIME ON NETWORK
  WALKTIME=DISTANCE/2.5*60
  _ATCNT[ATYPE]=_ATCNT[ATYPE]+1
  _FTCNT[FTYPE]=_FTCNT[FTYPE]+1
; Put Bike Speed and Time on network
  _spd_red=0
  _ln_red=0
    if (SPEED>12)
       _spd_red=(SPEED-12)/18
    endif
    if (Lanes=2)
       _ln_red=1
    endif
    if (Lanes>=2)
       _ln_red=2
    endif
    BK_SPD=12 - _spd_red - _ln_red
```

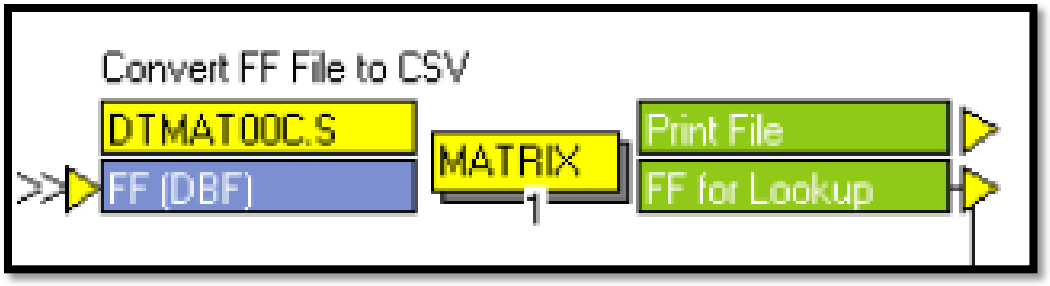
```
    if (BK_LNS=1,3) BK_SPD=12
    if (BK_LNS=2) BK_SPD=MAX(BK_SPD,11)
    if (FTYPE1==5) BK_SPD=12
    BK_TIME=60*DISTANCE/BK_SPD
If (FTYPE1==0) DELETE

ENDPROCESS

PROCESS  PHASE=SUMMARY
; Use this phase for combining and reporting of working variables.

ENDPROCESS
ENDRUN
```



HNHWY00A.S

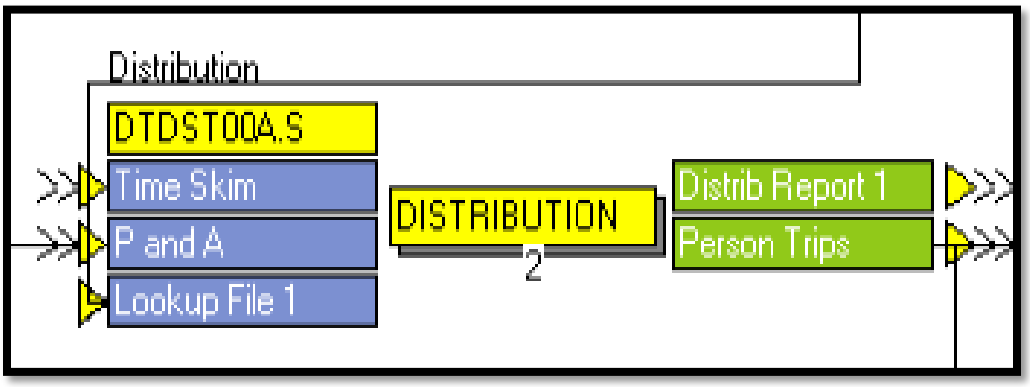
```
; Do not change filenames or add or remove FILEI/FILEO statements using an editor. Use
Cube/Application Manager.
RUN PGM=HIGHWAY PRNFILE="{SCENARIO_DIR}\Output\HNHWY00A.PRN" MSG='Travel Time Skim'
FILEI NETI = "{SCENARIO_DIR}\OUTPUT\UNLOADED.NET"
FILEO MATO[1] = "{SCENARIO_DIR}\OUTPUT\FHSKIMS.{ALT}{YEAR}.MAT",
 MO=1-2,10,99,3 NAME=TIME,DISTANCE,TERMINALTIME,WALKDISTANCE,BIKETIME, DEC=4*3
FILEI TURNPENI = "{SCENARIO_DIR}\INPUT\TCARDS.PEN"
ARRAY TERM=59 TERMTIME={ZONESA}
PAR ZONEMSG=100
TERM[1]={TERM10}
TERM[2]={TERM20}
TERM[3]={TERM30}
TERM[4]={TERM40}
TERM[5]={TERM50}
PROCESS PHASE=LINKREAD
   IF (A=1-{ZONESA}) TERMTIME[A]=TERM[LI.ATYPE1]
   IF (LI.FTYPE=10-19,70-99) ADDTOGROUP=1 ; no walk on freeways, etc.
   IF (LI.FTYPE=49) ADDTOGROUP=2 ; no autos allowed on FTYPE=49
ENDPROCESS
PROCESS PHASE=ILOOP
  PATHLOAD PATH=LI.TIME,EXCLUDEGROUP=2,
      MW[1]=PATHTRACE(LI.TIME,1),NOACCESS=99999,
      MW[2]=PATHTRACE(LI.DISTANCE),NOACCESS=99999, PENI=1
    PATHLOAD PATH=LI.BK_TIME,EXCLUDEGROUP=1,
      MW[3]=PATHTRACE(LI.BK_TIME),NOACCESS=99999
  MW[1][I]=LOWEST(1,2)/4 ; INTRAZONAL TIME = 1/2 THE AVERAGE OF THE TWO NEAREST ZONES
  MW[2][I]=LOWEST(2,2)/4 ; INTRAZONAL DISTANCE = 1/2 THE AVERAGE OF THE TWO NEAREST ZONES
  MW[3][I]=LOWEST(3,2)/4 ; INTRAZONAL BIKE TIME = 1/2 THE AVERAGE OF THE TWO NEAREST ZONES
  MW[10]=TERMTIME[I]+TERMTIME[J] ; BUILDS TERMINAL TIME MATRIX
  PATHLOAD PATH=LI.DISTANCE,EXCLUDEGROUP=1, MW[99]=PATHTRACE(LI.DISTANCE)
  MW[99][I]=ROWMIN(99)/2
 ENDPROCESS
PROCESS PHASE=ADJUST
ENDPROCESS
ENDRUN
```

Trip Distribution Step



DTMAT00C.S

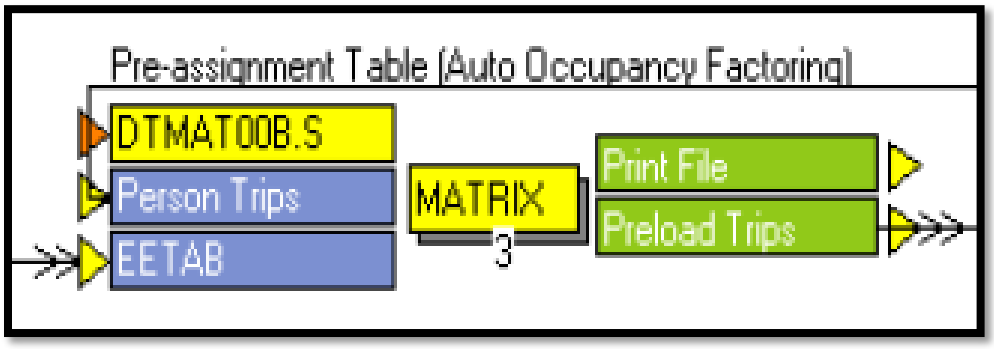
```
; Do not change filenames or add or remove FILEI/FILEO statements using an editor. Use
Cube/Application Manager.
RUN PGM=MATRIX PRNFILE="{SCENARIO_DIR}\Output\DTMAT00A.PRN" MSG='Convert FF File to CSV'
FILEO PRINTO[1] = "{SCENARIO_DIR}\OUTPUT\FF.CSV"
FILEI RECI = "{CATALOG_DIR}\PARAMETERS\FF.DBF"

print                                                                                csv=t,
list=ri.time,ri.hbwff,ri.hbshff,ri.hbsrff,ri.hboff,ri.nhbff,ri.tk4ff,ri.tksglff,ri.tktrlrff,
                ri.sovieff,ri.hovieff,ri.tkltieff,ri.tkhtieff,ri.hbuff,ri.hdormuff, printo=1
ENDRUN
```



DTDST00A.S

```
; Do not change filenames or add or remove FILEI/FILEO statements using an editor. Use
Cube/Application Manager.
RUN PGM=DISTRIBUTION PRNFILE="{SCENARIO_DIR}\output\DISTRIB.PRN" MSG='Distribution'
FILEO MATO[1] = "{SCENARIO_DIR}\Output\PTRIPS.MAT",
 MO=1-14,
NAME=HBW,HBSH,HBSR,HBO,NHB,TRUCK4,TRUCKSU,TRUCKTRLR,SOVIE,HOVIE,TRUCKLDIE,TRUCKHDIE,HBU,HDORMU
FILEI ZDATI[1] = "{SCENARIO_DIR}\output\PANDA.DBF"
FILEI MATI[1] = "{SCENARIO_DIR}\OUTPUT\FHSKIMS.{ALT}{YEAR}.MAT"
FILEI LOOKUPI[1] = "{SCENARIO_DIR}\OUTPUT\FF.CSV"
PAR ZONEMSG=100,  MAXRMSE=.001, MAXITERS=50

setpa p[1]=hbwp,a[1]=hbwa
setpa p[2]=hbshp,a[2]=hbsha
setpa p[3]=hbsrp,a[3]=hbsra
setpa p[4]=hbop,a[4]=hboa
setpa p[5]=nhbp,a[5]=nhba
setpa p[6]=tk4p,a[6]=tk4a
setpa p[7]=sglunitp,a[7]=sglunita
setpa p[8]=trktrlrp,a[8]=trktrlra
setpa p[9]=soviep,a[9]=soviea
setpa p[10]=hoviep,a[10]=hoviea
setpa p[11]=ldtkiep,a[11]=ldtkiea
setpa p[12]=hdtkiep,a[12]=hdtkiea
setpa p[13]=hbup,a[13]=hbua
setpa p[14]=hdormup,a[14]=hdormua
MW[50]=MI.1.TIME+MI.1.TERMINALTIME
```

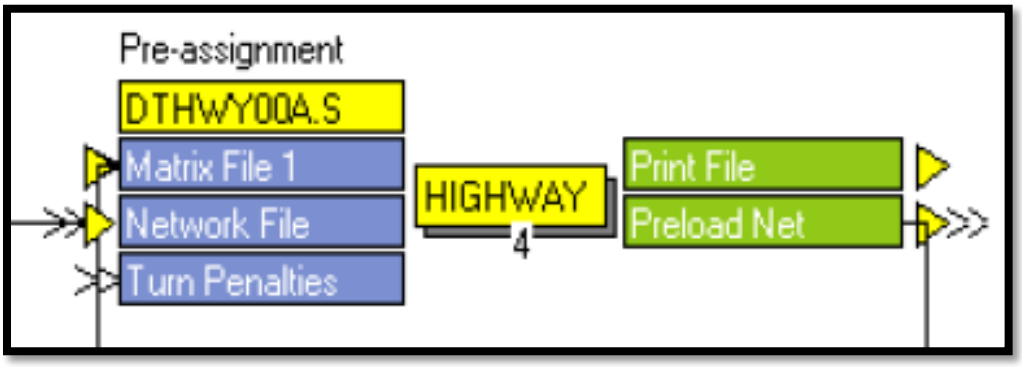
```
LOOKUP, NAME=FF,
   LOOKUP[1]=1, RESULT=2,
   LOOKUP[2]=1, RESULT=3,
   LOOKUP[3]=1, RESULT=4,
   LOOKUP[4]=1, RESULT=5,
   LOOKUP[5]=1, RESULT=6,
   LOOKUP[6]=1, RESULT=7,
   LOOKUP[7]=1, RESULT=8,
   LOOKUP[8]=1, RESULT=9,
   LOOKUP[9]=1, RESULT=10,
   LOOKUP[10]=1, RESULT=11,
   LOOKUP[11]=1, RESULT=12,
   LOOKUP[12]=1, RESULT=13,
   LOOKUP[13]=1, RESULT=14,
   LOOKUP[14]=1, RESULT=15,
   INTERPOLATE=Y, lookupi=1
GRAVITY LOS=MW[50], PURPOSE=1, FFACTORS=FF ; HBW
GRAVITY LOS=MW[50], PURPOSE=2, FFACTORS=FF ; HBSH
GRAVITY LOS=MW[50], PURPOSE=3, FFACTORS=FF ; HBSR
GRAVITY LOS=MW[50], PURPOSE=4, FFACTORS=FF ; HBO
GRAVITY LOS=MW[50], PURPOSE=5, FFACTORS=FF ; NHB
GRAVITY LOS=MW[50], PURPOSE=6, FFACTORS=FF ; TK4
GRAVITY LOS=MW[50], PURPOSE=7, FFACTORS=FF ; SGLUNIT
GRAVITY LOS=MW[50], PURPOSE=8, FFACTORS=FF ; TRKTLR
GRAVITY LOS=MW[50], PURPOSE=9, FFACTORS=FF ; SOVIE
GRAVITY LOS=MW[50], PURPOSE=10, FFACTORS=FF ; HOVIE
GRAVITY LOS=MW[50], PURPOSE=11, FFACTORS=FF ; LDTKIE
GRAVITY LOS=MW[50], PURPOSE=12, FFACTORS=FF ; HDTKIE
GRAVITY LOS=MW[50], PURPOSE=13, FFACTORS=FF ; HBU
GRAVITY LOS=MW[50], PURPOSE=14, FFACTORS=FF ; HDORMU

FREQUENCY BASEMW=50,VALUEMW= 1,RANGE=0-60-5.0, TITLE='HBW TLFD'
FREQUENCY BASEMW=50,VALUEMW= 2,RANGE=0-60-5.0, TITLE='HBSH TLFD'
FREQUENCY BASEMW=50,VALUEMW= 3,RANGE=0-60-5.0, TITLE='HBSR TLFD'
FREQUENCY BASEMW=50,VALUEMW= 4,RANGE=0-60-5.0, TITLE='HBO TLFD'
FREQUENCY BASEMW=50,VALUEMW= 5,RANGE=0-60-5.0, TITLE='NHB TLFD'
FREQUENCY BASEMW=50,VALUEMW= 6,RANGE=0-60-5.0, TITLE='TK4 TLFD'
FREQUENCY BASEMW=50,VALUEMW= 7,RANGE=0-60-5.0, TITLE='SGLUNIT TLFD'
FREQUENCY BASEMW=50,VALUEMW= 8,RANGE=0-60-5.0, TITLE='TRKTLR TLFD'
FREQUENCY BASEMW=50,VALUEMW= 9,RANGE=0-60-5.0, TITLE='SOVIE TLFD'
FREQUENCY BASEMW=50,VALUEMW=10,RANGE=0-60-5.0, TITLE='HOVIE TLFD'
FREQUENCY BASEMW=50,VALUEMW=11,RANGE=0-60-5.0, TITLE='LDTKIE TLFD'
FREQUENCY BASEMW=50,VALUEMW=12,RANGE=0-60-5.0, TITLE='HDTKIE TLFD'
FREQUENCY BASEMW=50,VALUEMW=13,RANGE=0-60-5.0, TITLE='HBU TLFD'
FREQUENCY BASEMW=50,VALUEMW=14,RANGE=0-60-5.0, TITLE='HDORMU TLFD'

ENDRUN
```



DTMAT00B.S

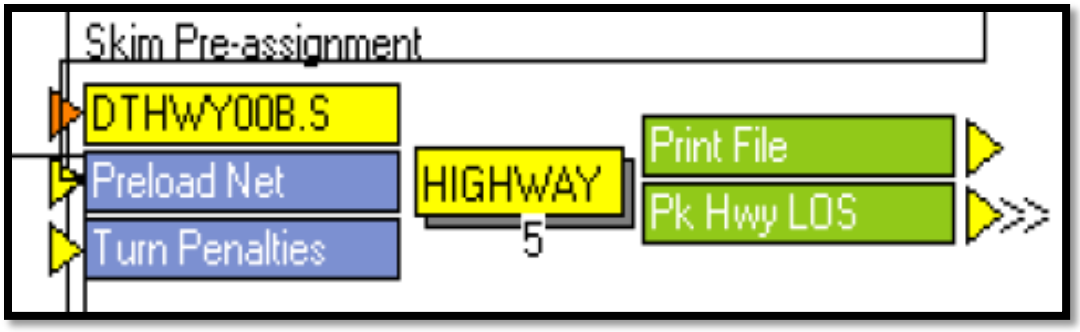
```
; Do not change filenames or add or remove FILEI/FILEO statements using an editor. Use
Cube/Application Manager.
RUN PGM=MATRIX PRNFILE="{SCENARIO_DIR}\Output\DTMAT00B.PRN" MSG='Pre-assignment Table (Auto
Occupancy Factoring)'
```

```
FILEI MATI[2] = "{SCENARIO_DIR}\output\EETAB.MAT"
FILEI MATI[1] = "{SCENARIO_DIR}\Output\PTRIPS.MAT"
FILEO MATO[1] = "{SCENARIO_DIR}\output\HTTAB.TEM.MAT",
 MO=1, NAME=PRELOADVEH

; The MATRIX module does not have any explicit phases.  The module does run within an implied ILOOP
; where I is the origin zones.  All user statements in the module are processed once for each
origin.
; Matrix computation (MW[#]=) are solved for all values of J for each I.  Thus for a given origin
zone I
; the values for all destination zones J are automatically computed.  The user can control the
computations
; at each J by using a JLOOP.
PAR ZONEMSG=100
MW[1]=(MI.1.1+MI.1.1.T)*0.5*{AOFAC1}+
(MI.1.2+MI.1.2.T)*0.5*{AOFAC2}+
(MI.1.3+MI.1.3.T)*0.5*{AOFAC3}+
(MI.1.4+MI.1.4.T)*0.5*{AOFAC4}+
(MI.1.5+MI.1.5.T)*0.5*{AOFAC1}+
(MI.1.6+MI.1.6.T)*0.5+
(MI.1.7+MI.1.7.T)*0.5+
(MI.1.8+MI.1.8.T)*0.5+
(MI.1.9+MI.1.9.T)*0.5+
(MI.1.10+MI.1.10.T)*0.5+
(MI.1.11+MI.1.11.T)*0.5+
(MI.1.12+MI.1.12.T)*0.5+
 mi.2.EETRIPS+
(MI.1.13+MI.1.13.T)*0.5*{AOFACU}  ; HBU
;(MI.1.14+MI.1.14.T)*0.5   ; HDORMU - don't include here because these are mostly not auto.

ENDRUN
```



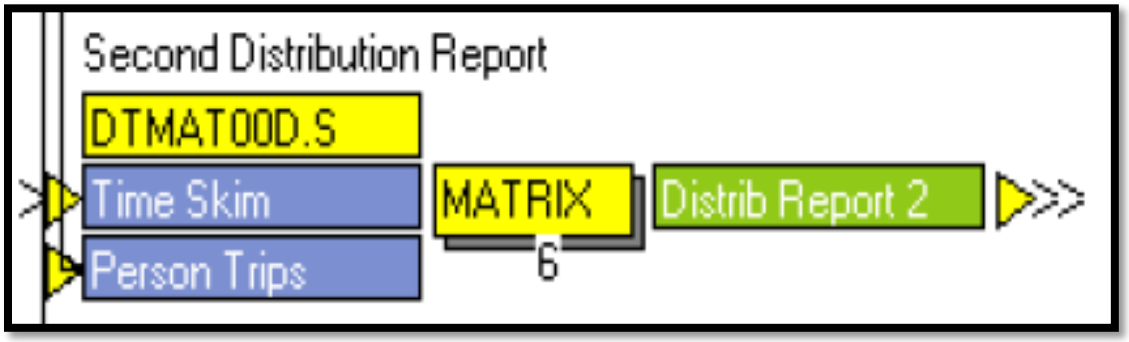
DTHWY00A.S

```
; Do not change filenames or add or remove FILEI/FILEO statements using an editor. Use
Cube/Application Manager.
RUN PGM=HIGHWAY PRNFILE="{SCENARIO_DIR}\Output\DTHWY00A.PRN" MSG='Pre-assignment'
FILEI NETI = "{SCENARIO_DIR}\OUTPUT\UNLOADED.NET"
FILEO NETO = "{SCENARIO_DIR}\OUTPUT\PRELOAD.NET"
FILEI TURNPENI = "{SCENARIO_DIR}\input\TCARDS.PEN"
FILEI MATI[1] = "{SCENARIO_DIR}\output\HTTAB.TEM.MAT"

PAR ZONEMSG=100,  MAXITERS=50

PROCESS PHASE=LINKREAD
; USE THE USER SUPPLIED ALPHA AND BETA FOR THE BPR CURVE
  IF (LI.BPRCOEFFICIENT=0)
     LW.BPRCOEFFICIENT=0.15
  ELSE
     LW.BPRCOEFFICIENT=LI.BPRCOEFFICIENT
  ENDIF
  IF (LI.BPREXPONENT=0)
     LW.BPREXPONENT=4.0
  ELSE
     LW.BPREXPONENT=LI.BPREXPONENT
  ENDIF
  IF (LI.CAPACITY=0)
     LW.DAILYCAP=999999
  ELSE
      LW.DAILYCAP=(LI.CAPACITY/li.confac)*li.uroadfactor
  ENDIF
  IF (LI.TIME=0)
     LW.FFTIME=0.00001
  ELSE
     LW.FFTIME=LI.TIME
  ENDIF
C=LW.DAILYCAP
T0=LW.FFTIME
IF (LI.FTYPE=49) ADDTOGROUP=1

ENDPROCESS

PROCESS PHASE=ILOOP
  MW[1]=MI.1.PRELOADVEH
  PATHLOAD PATH=TIME, VOL[1]=MW[1],EXCLUDEGROUP=1,PENI=1
ENDPROCESS

PROCESS PHASE=ADJUST
  FUNCTION   TC[1]=T0*(1+LW.BPRCOEFFICIENT*(V/C)^LW.BPREXPONENT) ; congested time equation, no toll
model in place
ENDPROCESS

ENDRUN
```



DTHWY00B.S

```
; Do not change filenames or add or remove FILEI/FILEO statements using an editor. Use
Cube/Application Manager.
RUN PGM=HIGHWAY PRNFILE="{SCENARIO_DIR}\Output\DTHWY00B.PRN" MSG='Skim Pre-assignment'
FILEI NETI = "{SCENARIO_DIR}\OUTPUT\PRELOAD.NET"
FILEO MATO[1] = "{SCENARIO_DIR}\OUTPUT\RHSKIMS.MAT",
 MO=1-2,10, NAME=TIME,DISTANCE,TERMINALTIME
FILEI TURNPENI = "{SCENARIO_DIR}\input\TCARDS.PEN"
ARRAY TERM=59 TERMTIME={ZONESA}
PAR ZONEMSG=100
TERM[1]={TERM10}
TERM[2]={TERM20}
TERM[3]={TERM30}
TERM[4]={TERM40}
TERM[5]={TERM50}
PROCESS PHASE=LINKREAD
   IF (A=1-{ZONESA}) TERMTIME[A]=TERM[LI.ATYPE1] ; BUILDS TERMINAL TIME ARRAY (KDK fixed again)
   IF (LI.FTYPE=49) ADDTOGROUP=1 ; no autos allowed on FTYPE=49
ENDPROCESS
PROCESS PHASE=ILOOP
  PATHLOAD PATH=LI.TIME, EXCLUDEGROUP=1,
      MW[1]=PATHTRACE(LI.TIME_1,1),NOACCESS=99999,
      MW[2]=PATHTRACE(LI.DISTANCE),NOACCESS=99999, PENI=1
  MW[1][I]=LOWEST(1,2)/4 ; INTRAZONAL TIME = 1/2 THE AVERAGE OF THE TWO NEAREST ZONES
  MW[2][I]=LOWEST(2,2)/4 ; INTRAZONAL DISTANCE = 1/2 THE AVERAGE OF THE TWO NEAREST ZONES
  MW[10]=TERMTIME[I]+TERMTIME[J] ; BUILDS TERMINAL TIME MATRIX
ENDPROCESS
PROCESS PHASE=ADJUST
ENDPROCESS
ENDRUN
```

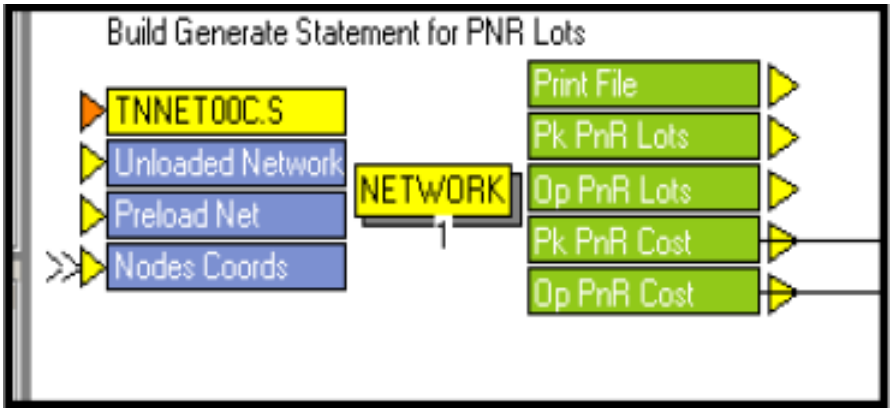


DTMAT00D.S

```
; Do not change filenames or add or remove FILEI/FILEO statements using an editor. Use
Cube/Application Manager.
RUN PGM=MATRIX PRNFILE="{SCENARIO_DIR}\output\DISTRIB2.PRN" MSG='Second Distribution Report'
FILEI MATI[2] = "{SCENARIO_DIR}\Output\PTRIPS.MAT"
FILEI MATI[1] = "{SCENARIO_DIR}\OUTPUT\FHSKIMS.{ALT}{YEAR}.MAT"
```

```
MW[50]=MI.1.TIME+MI.1.TERMINALTIME
MW[1] = MW[50]*MI.2.HBW
MW[2] = MW[50]*MI.2.HBSH
MW[3] = MW[50]*MI.2.HBSR
MW[4] = MW[50]*MI.2.HBO
MW[5] = MW[50]*MI.2.NHB
MW[6] = MW[50]*MI.2.TRUCK4
MW[7] = MW[50]*MI.2.TRUCKSU
MW[8] = MW[50]*MI.2.TRUCKTRLR
MW[9] = MW[50]*MI.2.SOVIE
MW[10]= MW[50]*MI.2.HOVIE
MW[11]= MW[50]*MI.2.TRUCKLDIE
MW[12]= MW[50]*MI.2.TRUCKHDIE
MW[13]= MW[50]*MI.2.HBU
MW[14]= MW[50]*MI.2.HDORMU

ENDRUN
```

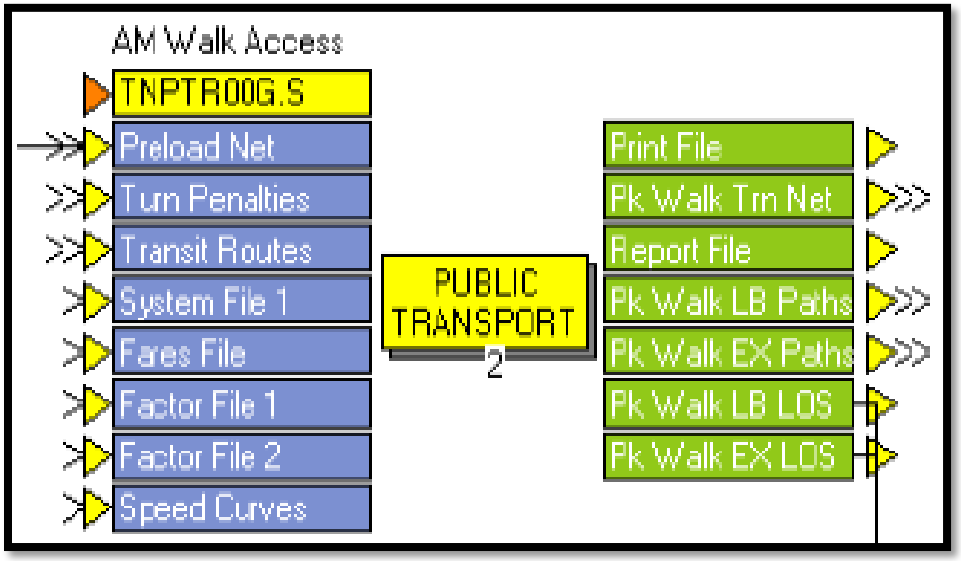
Transit Network Step



TNNET00C.S

```
; Do not change filenames or add or remove FILEI/FILEO statements using an editor. Use
Cube/Application Manager.
RUN PGM=NETWORK PRNFILE="{SCENARIO_DIR}\output\TNNET00A.PRN" MSG='Build Generate Statement for PNR
Lots'
FILEI LOOKUPI[1] = "{SCENARIO_DIR}\output\NODECOOR.csv"
FILEI LINKI[2] = "{SCENARIO_DIR}\output\PRELOAD.NET"
FILEI LINKI[1] = "{SCENARIO_DIR}\output\UNLOADED.NET"
FILEO PRINTO[4] = "{SCENARIO_DIR}\output\OPPNRCOST.CSV"
FILEO PRINTO[3] = "{SCENARIO_DIR}\output\PKPNRCOST.CSV"
FILEO PRINTO[2] = "{SCENARIO_DIR}\output\MD_STATDATA.CSV"
FILEO PRINTO[1] = "{SCENARIO_DIR}\output\AM_STATDATA.CSV"
ARRAY STATSTOP=99999 STATNUMB=99999, statspaces=99999, PNRTERM=99999, KNRTERM=99999, nrz=99999
; add in nearest centroid lookup for auto cost to stations HWYOPCOST

PROCESS PHASE=NODEMERGE
; put nodes, x and y coordinates into memory for lookup nearest TAZ question
lookup lookupi=1,name=netcoord, lookup[1]=1, result=2, lookup[2]=1, result=3, fail=0
; extract am station info from network for later calculations
IF (AMUSEFLAG=1)

workstat=N
workstatx=netcoord(1,workstat,0)
workstaty=netcoord(2,workstat,0)
mindist=999.99
loop _ww=1,{ZONESA}
  zx=netcoord(1,_ww,0)
  zy=netcoord(2,_ww,0)
  if (_ww!=workstat) dist=sqrt((workstatx-zx)^2+(workstaty-zy)^2)/{units}
  if (dist<mindist) mindist=dist, nearestzone=_ww
endloop
```

```
 PRINT  form=5.0,list="GENERATE,COST=(li.distance),MINCOST=12*1.0,MAXCOST=12*",  pnrsvcarea(5.2L),
",EXTRACTCOST=(li.TIME_1),LIST=F,DIRECTION=1,NTLEGMODE=2,  FROMNODE=1-{ZONESA},TONODE=",N,PRINTO=1
 PRINT CSV=T,  LIST=N(6.0),AMPNRCOST,NEARESTZONE(6.0)  PRINTO=3
endif
; extract md station info from network for later calculations
IF (MDUSEFLAG=1)

workstat=N
workstatx=netcoord(1,workstat,0)
workstaty=netcoord(2,workstat,0)
mindist=999.99
loop _ww=1,{ZONESA}
  zx=netcoord(1,_ww,0)
  zy=netcoord(2,_ww,0)
  if (_ww!=workstat) dist=sqrt((workstatx-zx)^2+(workstaty-zy)^2)/{units}
  if (dist<mindist) mindist=dist, nearestzone=_ww
endloop

 PRINT                        form=5.0,list="GENERATE,COST=(li.distance),MINCOST=12*1.0,MAXCOST=12*"
,pnrsvcarea(5.2L),",EXTRACTCOST=(li.TIME),LIST=F,DIRECTION=1,NTLEGMODE=2,                FROMNODE=1-
{ZONESA},TONODE=",N,PRINTO=2
 PRINT CSV=T,  LIST=N(6.0),MDPNRCOST,NEARESTZONE(6.0)  PRINTO=4
endif

ENDPROCESS
ENDRUN
```

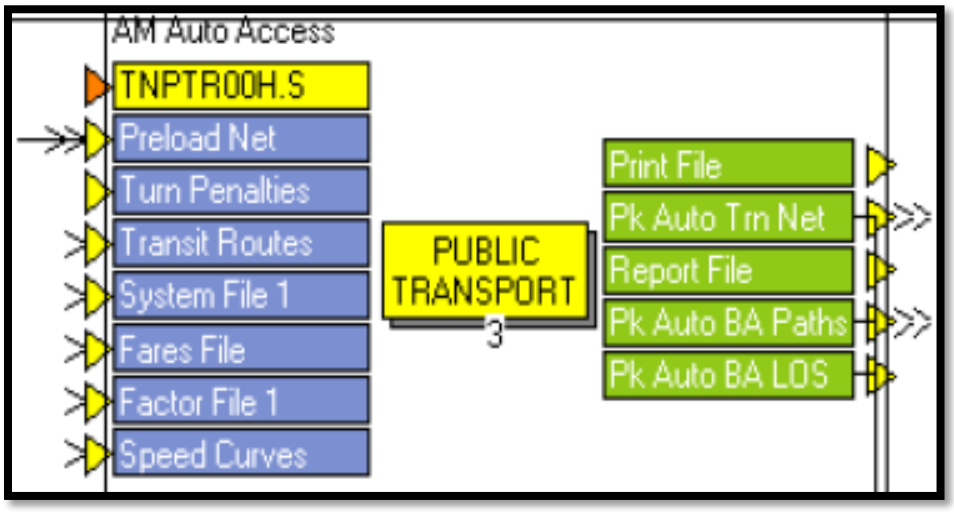


TNPTR00G.S

```
; Script for program PUBLIC TRANSPORT in file "C:\FSUTMS\DISTRICT2\ALACHUA\TNPTR00C.S"
;  Do  not  change  filenames  or  add  or  remove  FILEI/FILEO  statements  using  an  editor.  Use
Cube/Application Manager.
RUN PGM=PUBLIC TRANSPORT PRNFILE="{SCENARIO_DIR}\output\TNPTR00D.PRN" MSG='AM Walk Access'
FILEI NETI = "{SCENARIO_DIR}\OUTPUT\PRELOAD.NET"
FILEI LINEI[1] = "{SCENARIO_DIR}\input\troute20{YEAR}.lin"
FILEI TURNPENI = "{SCENARIO_DIR}\input\TCARDS.PEN"
FILEI LOOKUPI[1] = "{CATALOG_DIR}\PARAMETERS\SPDCRV.CSV"
FILEO MATO[2] = "{SCENARIO_DIR}\output\WALKPREMAM.MAT",
 MO=1-11, NAME=TIME1M,TIME2M,TIME3M,TIME4M,TIME6M,TIME8M,IWAIT,XWAIT,IVTT,OVTT,FARE
FILEO MATO[1] = "{SCENARIO_DIR}\output\WALKAM.MAT",
 MO=1-11, NAME=TIME1M,TIME2M,TIME3M,TIME4M,TIME6M,TIME8M,IWAIT,XWAIT,IVTT,OVTT,FARE
FILEO REPORTO = "{SCENARIO_DIR}\output\TNPTR00C.PRN"
FILEI FACTORI[2] = "{CATALOG_DIR}\Parameters\ALACHUAWKPREM.FAC"
FILEI FACTORI[1] = "{CATALOG_DIR}\PARAMETERS\ALACHUAWLB.FAC"
```

```
FILEI FAREI = "{CATALOG_DIR}\parameters\ALACHUA.FAR"
FILEO ROUTEO[2] = "{SCENARIO_DIR}\output\WALKPREMAM.RTE",
 REPORTI=1-{zonesa}, REPORTJ={cbdzone}
FILEO ROUTEO[1] = "{SCENARIO_DIR}\output\WALKLBAM.RTE",
 REPORTI=1-{zonesa}, REPORTJ={cbdzone}
FILEI SYSTEMI = "{CATALOG_DIR}\PARAMETERS\ALACHUA.PTS"
FILEO NETO = "{SCENARIO_DIR}\output\TNETWALKAM.NET"
PARAMETERS TRANTIME=(LI.TIME_1*1.5), ;GENERIC TRANSIT VS. AUTO RUN TIME
           TRANTIME[4]=LW.LBTIME,    ; MODE SPECIFIC TRANSIT VS. AUTO RUN TIME
           TRANTIME[6]=LW.EBTIME,    ; CREATED THROUGH A LOOKUP FUNCTION
           TRANTIME[8]=LW.RLTIME,
           FARE=F, USERCLASSES=1-2,  HDWAYPERIOD=1  MAPSCALE={UNITS}
REPORT LINES=T
PROCESS PHASE=LINKREAD
   LW.WALKTIME=60*LI.DISTANCE/{walkspeed}
   IF (LI.TIME_1>0) LW.SPEED=60*LI.DISTANCE/LI.TIME_1
   LOOKUP, NAME=CURVES, LOOKUP[1]=1, RESULT=2, LOOKUP[2]=1, RESULT=3, LOOKUP[3]=1, RESULT=4,
           INTERPOLATE=Y, LOOKUPI=1
   IF (LI.FTYPE=10-19,80-99) ; FREE FLOW CONDITIONS FOR ALL
       LW.LBCURVE=1,LW.EBCURVE=1,LW.RLCURVE=1
   ELSEIF (LI.FTYPE=20-79&LI.ATYPE=10-19) ; BUSSES HITTING RESISTANCE, RAIL NO CONFLICTS (GRADE
SEP)
       LW.LBCURVE=2,LW.EBCURVE=2,LW.RLCURVE=1
   ELSEIF (LI.FTYPE=20-79&LI.ATYPE=20-29) ; BUSSES HITTING RESISTANCE(LB MORE), RAIL NO CONFLICTS
       LW.LBCURVE=3,LW.EBCURVE=2,LW.RLCURVE=1
   ELSEIF (LI.FTYPE=20-79&LI.ATYPE=30-39) ; BUSSES HITTING RESISTANCE(LB MORE), RAIL NO CONFLICTS
       LW.LBCURVE=3,LW.EBCURVE=2,LW.RLCURVE=1
   ELSEIF (LI.FTYPE=20-79&LI.ATYPE=40-49) ; BUSSES HITTING RESISTANCE(LB MORE), RAIL NO CONFLICTS
       LW.LBCURVE=2,LW.EBCURVE=2,LW.RLCURVE=1
   ELSEIF (LI.FTYPE=20-79&LI.ATYPE=50-59) ; BUSSES HITTING RESISTANCE(LB MORE), RAIL NO CONFLICTS
       LW.LBCURVE=2,LW.EBCURVE=1,LW.RLCURVE=1
   ENDIF
       LW.LBSPEED=CURVES(LW.LBCURVE,LW.SPEED)
       LW.EBSPEED=CURVES(LW.EBCURVE,LW.SPEED)
       LW.RLSPEED=CURVES(LW.RLCURVE,LW.SPEED)
       LW.LBTIME=60*LI.DISTANCE/LW.LBSPEED
       LW.EBTIME=60*LI.DISTANCE/LW.EBSPEED
       LW.RLTIME=60*LI.DISTANCE/LW.RLSPEED
       TRANTIME[4]=LW.LBTIME
       TRANTIME[6]=LW.EBTIME
       TRANTIME[8]=LW.RLTIME
   ENDPROCESS

PROCESS PHASE=DATAPREP
; WALK ACCESS
  GENERATE, COST=(LW.WALKTIME),MAXCOST=103*24.0,LIST=T,NTLEGMODE = 1,
            DIRECTION=1, FROMNODE=1-{zonesa}, TONODE=1000-99999
; WALK EGRESS
  GENERATE, COST=(LW.WALKTIME),MAXCOST=103*24.0,LIST=T,NTLEGMODE=101,
            DIRECTION=2, FROMNODE=1-{zonesa}, TONODE=1000-99999
; WALK CONNECTORS
  GENERATE, COST=(LW.WALKTIME),MAXCOST=103*12,LIST=T,NTLEGMODE = 3,DIRECTION=3,
            FROMNODE=1000-99999, TONODE=1000-99999
ENDPROCESS

PROCESS PHASE=SKIMIJ
  MW[1]=TIMEA(0,1,101)
  MW[2]=TIMEA(0,2,102)
  MW[3]=TIMEA(0,3)
  MW[4]=TIMEA(0,4)
  MW[5]=TIMEA(0,6)
  MW[6]=TIMEA(0,8)
  MW[7]=IWAITA(0)
  MW[8]=XWAITA(0)
  MW[9]=TIMEA(0,TMODES)
  MW[10]=TIMEA(0,NTMODES)
  MW[11]=FAREA(0,ALLMODES)

;VARIOUS THINGS THAT CAN BE SKIMMED
/*
  COMPCOST(RouteSet)       Skims Composite Costs
```

```
  ValOfChoice(RouteSet)     Skims Value of Choice
  IWAITA(RouteSet)          Skims Initial Wait Times Actual
  XWAITA(RouteSet)          Skims Transfer Wait Times Actual
  IWAITP(RouteSet)          Skims Initial Wait Times Perceived
  XWAITP(RouteSet)          Skims Initial Transfer Times Perceived
  TIMEA(RouteSet, Mode)     Skims Travel Time Actual
  TIMEP(RouteSet, Mode)     Skims Travel Time Perceived
  XFERPENA(RouteSet, Mode)  Skims Transfer Penalty Actual
  XFERPENP(RouteSet, Mode)  Skims Transfer Penalty Actual
  DIST(RouteSet, Mode)      Skims Distance
  BRDINGS(RouteSet, Mode)   Skims Number of Boardings (xfers+1)
  BESTJRNY                  Skims Best Journey Times
  FAREA(RouteSet, Mode)     Skims Fares in Monetary units
  FAREP(RouteSet, Mode)     Skims Fares in Generalized Time units
*/
ENDPROCESS
ENDRUN
```

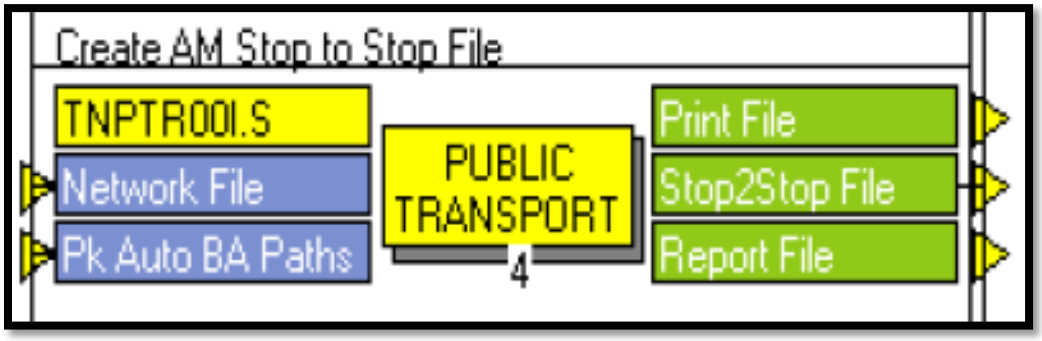


TNPTR00H.S

```
; Script for program PUBLIC TRANSPORT in file "C:\FSUTMS\DISTRICT2\ALACHUA\TNPTR00D.S"
; Do not change filenames or add or remove FILEI/FILEO statements using an editor. Use
Cube/Application Manager.
RUN PGM=PUBLIC TRANSPORT PRNFILE="{SCENARIO_DIR}\output\TNPTR00G.PRN" MSG='AM Auto Access'
FILEI NETI = "{SCENARIO_DIR}\OUTPUT\PRELOAD.NET"
FILEI LOOKUPI[1] = "{CATALOG_DIR}\PARAMETERS\SPDCRV.CSV"
FILEI TURNPENI = "{SCENARIO_DIR}\input\TCARDS.PEN"
FILEO REPORTO = "{SCENARIO_DIR}\output\TNPTR00F.PRN"
FILEO MATO[1] = "{SCENARIO_DIR}\output\AUTOAM.MAT",
 MO=1-11, NAME=TIME1M,TIME2M,TIME3M,TIME4M,TIME6M,TIME8M,IWAIT,XWAIT,IVTT,OVTT,FARE
FILEI FACTORI[1] = "{CATALOG_DIR}\PARAMETERS\ALACHUAPNR.FAC"
FILEI FAREI = "{CATALOG_DIR}\parameters\ALACHUA.FAR"
FILEO ROUTEO[1] = "{SCENARIO_DIR}\output\AUTOALLAM.RTE",
 REPORTI=1-{zonesa}, REPORTJ={cbdzone}
FILEI SYSTEMI = "{CATALOG_DIR}\PARAMETERS\ALACHUA.PTS"
FILEO NETO = "{SCENARIO_DIR}\output\TNETAUTOAM.NET"
FILEI LINEI[1] = "{SCENARIO_DIR}\input\troute20{YEAR}.lin"
PARAMETERS TRANTIME=(LI.TIME_1*1.5), ;GENERIC TRANSIT VS. AUTO RUN TIME
           TRANTIME[4]=LW.LBTIME,    ; MODE SPECIFIC TRANSIT VS. AUTO RUN TIME
           TRANTIME[6]=LW.EBTIME,    ; CREATED THROUGH A LOOKUP FUNCTION
           TRANTIME[8]=LW.RLTIME,
           FARE=F, USERCLASSES=1,    MAPSCALE={UNITS}

           HDWAYPERIOD=1

REPORT LINES=T
```

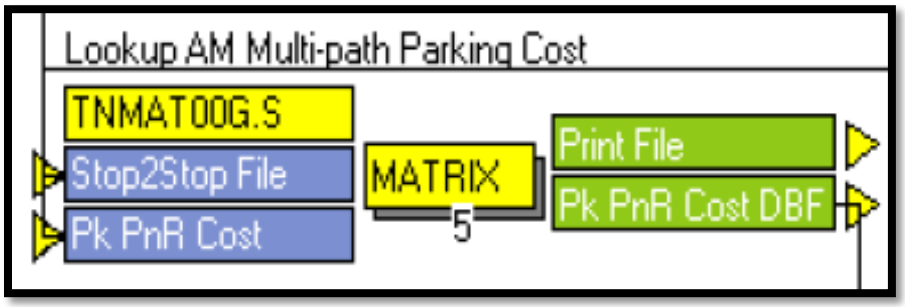
```
PROCESS PHASE=LINKREAD
   LW.WALKTIME=60*LI.DISTANCE/{walkspeed}
   IF (LI.TIME_1>0) LW.SPEED=60*LI.DISTANCE/LI.TIME_1
   LOOKUP, NAME=CURVES, LOOKUP[1]=1, RESULT=2, LOOKUP[2]=1, RESULT=3, LOOKUP[3]=1, RESULT=4,
           INTERPOLATE=Y, LOOKUPI=1
   IF (LI.FTYPE=10-19,80-99) ; FREE FLOW CONDITIONS FOR ALL
       LW.LBCURVE=1,LW.EBCURVE=1,LW.RLCURVE=1
   ELSEIF (LI.FTYPE=20-79&LI.ATYPE=10-19) ; BUSSES HITTING RESISTANCE, RAIL NO CONFLICTS (GRADE
SEP)
       LW.LBCURVE=2,LW.EBCURVE=2,LW.RLCURVE=1
   ELSEIF (LI.FTYPE=20-79&LI.ATYPE=20-29) ; BUSSES HITTING RESISTANCE(LB MORE), RAIL NO CONFLICTS
       LW.LBCURVE=3,LW.EBCURVE=2,LW.RLCURVE=1
   ELSEIF (LI.FTYPE=20-79&LI.ATYPE=30-39) ; BUSSES HITTING RESISTANCE(LB MORE), RAIL NO CONFLICTS
       LW.LBCURVE=3,LW.EBCURVE=2,LW.RLCURVE=1
   ELSEIF (LI.FTYPE=20-79&LI.ATYPE=40-49) ; BUSSES HITTING RESISTANCE(LB MORE), RAIL NO CONFLICTS
       LW.LBCURVE=2,LW.EBCURVE=2,LW.RLCURVE=1
   ELSEIF (LI.FTYPE=20-79&LI.ATYPE=50-59) ; BUSSES HITTING RESISTANCE(LB MORE), RAIL NO CONFLICTS
       LW.LBCURVE=2,LW.EBCURVE=1,LW.RLCURVE=1
   ENDIF
       LW.LBSPEED=CURVES(LW.LBCURVE,LW.SPEED)
       LW.EBSPEED=CURVES(LW.EBCURVE,LW.SPEED)
       LW.RLSPEED=CURVES(LW.RLCURVE,LW.SPEED)
       LW.LBTIME=60*LI.DISTANCE/LW.LBSPEED
       LW.EBTIME=60*LI.DISTANCE/LW.EBSPEED
       LW.RLTIME=60*LI.DISTANCE/LW.RLSPEED
       TRANTIME[4]=LW.LBTIME
       TRANTIME[6]=LW.EBTIME
       TRANTIME[8]=LW.RLTIME

ENDPROCESS

PROCESS PHASE=DATAPREP
; AUTO ACCESS
READ,
 FILE = "{SCENARIO_DIR}\OUTPUT\AM_STATDATA.CSV"
; WALK EGRESS
  GENERATE, COST=(LW.WALKTIME),MAXCOST=103*24.0,LIST=T,NTLEGMODE=101,
            DIRECTION=2, FROMNODE=1-{zonesa}, TONODE=1000-99999
; WALK CONNECTORS
  GENERATE, COST=(LW.WALKTIME),MAXCOST=103*12,LIST=T,NTLEGMODE = 3,DIRECTION=3,
            FROMNODE=1000-99999, TONODE=1000-99999

ENDPROCESS

PROCESS PHASE=SKIMIJ
  MW[1]=TIMEA(0,1,101)
  MW[2]=TIMEA(0,2,102)
  MW[3]=TIMEA(0,3)
  MW[4]=TIMEA(0,4)
  MW[5]=TIMEA(0,6)
  MW[6]=TIMEA(0,8)
  MW[7]=IWAITA(0)
  MW[8]=XWAITA(0)
  MW[9]=TIMEA(0,TMODES)
  MW[10]=TIMEA(0,NTMODES)
  MW[11]=FAREA(0,ALLMODES)

;VARIOUS THINGS THAT CAN BE SKIMMED
/*
  COMPCOST(RouteSet)        Skims Composite Costs
  ValOfChoice(RouteSet)     Skims Value of Choice
  IWAITA(RouteSet)          Skims Initial Wait Times Actual
  XWAITA(RouteSet)          Skims Transfer Wait Times Actual
  IWAITP(RouteSet)          Skims Initial Wait Times Perceived
  XWAITP(RouteSet)          Skims Initial Transfer Times Perceived
  TIMEA(RouteSet, Mode)     Skims Travel Time Actual
  TIMEP(RouteSet, Mode)     Skims Travel Time Perceived
  XFERPENA(RouteSet, Mode)  Skims Transfer Penalty Actual
  XFERPENP(RouteSet, Mode)  Skims Transfer Penalty Actual
  DIST(RouteSet, Mode)      Skims Distance
  BRDINGS(RouteSet, Mode)   Skims Number of Boardings (xfers+1)
```

```
  BESTJRNY                  Skims Best Journey Times
  FAREA(RouteSet, Mode)     Skims Fares in Monetary units
  FAREP(RouteSet, Mode)     Skims Fares in Generalized Time units
*/

ENDPROCESS

ENDRUN
```



TNPTR00I.S

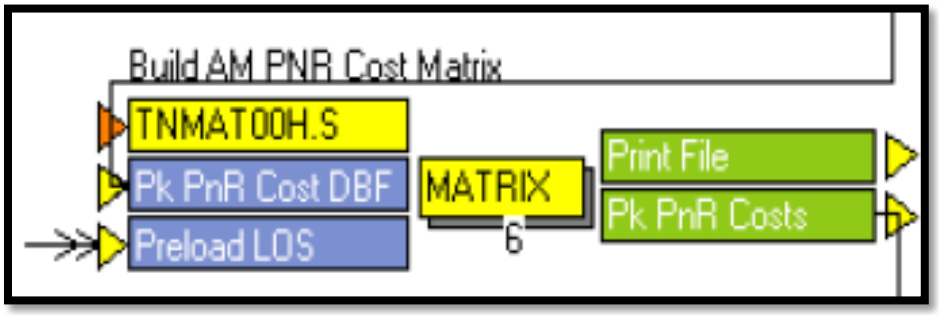
```
; Script for program PUBLIC TRANSPORT in file "C:\FSUTMS\DISTRICT2\ALACHUA\TNPTR00E.S"
; Do not change filenames or add or remove FILEI/FILEO statements using an editor. Use
Cube/Application Manager.
RUN PGM=PUBLIC TRANSPORT PRNFILE="{SCENARIO_DIR}\output\TNPTR00I.PRN" MSG='Create AM Stop to Stop
File'
FILEO REPORTO = "{SCENARIO_DIR}\output\TNPTR00J.PRN"
FILEO STOP2STOPO = "{SCENARIO_DIR}\output\AMPNR.DBF",
 ACCUMULATE=FIRSTLAST, NODES=1-99999
FILEI NETI = "{SCENARIO_DIR}\output\TNETAUTOAM.NET"
FILEI ROUTEI[1] = "{SCENARIO_DIR}\output\AUTOALLAM.RTE"
PARAMETERS HDWAYPERIOD=1,
           TRIPSIJ[1]=100,
           NOROUTEERRS=999999999

ENDRUN
```



TNMAT00G.S

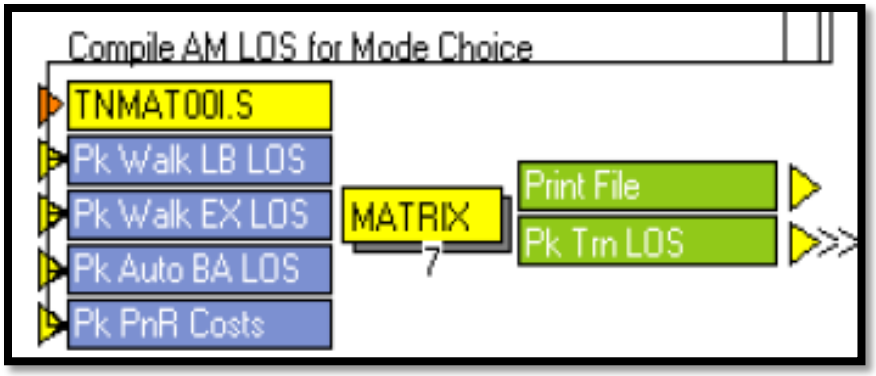
```
; Script for program MATRIX in file "C:\FSUTMS\DISTRICT2\ALACHUA\TNMAT00E.S"
; Do not change filenames or add or remove FILEI/FILEO statements using an editor. Use
Cube/Application Manager.
RUN PGM=MATRIX PRNFILE="{SCENARIO_DIR}\output\TNMAT00E.PRN" MSG='Lookup AM Multi-path Parking Cost'
FILEO RECO[1] = "{SCENARIO_DIR}\output\AMPCOST.DBF",
 FIELDS=ORZ,DSZ,MA,MEANCOST,MB,STNZONE,MC,CNT
FILEI RECI = "{SCENARIO_DIR}\output\AMPNR.DBF"
FILEI LOOKUPI[1] = "{SCENARIO_DIR}\output\PKPNRCOST.CSV"

LOOKUP, NAME=STATIONS, LOOKUP[1]=1, RESULT=2, LOOKUP[2]=1, RESULT=3, INTERPOLATE=F, FAIL[1]=0,
FAIL[2]=0, LOOKUPI=1
  RO.ORZ=RI.I
```

```
  RO.DSZ=RI.J
  statnode=ri.fromnode
  RO.MA=1
  RO.MB=2
  RO.MC=3
  RO.CNT=1
  PCOST=STATIONS(1,statnode)
  RO.STNZONE=STATIONS(2,STATNODE)
  meancost=PCOST*ri.vol/100
WRITE RECO=1
ENDRUN
```



TNMAT00H.S

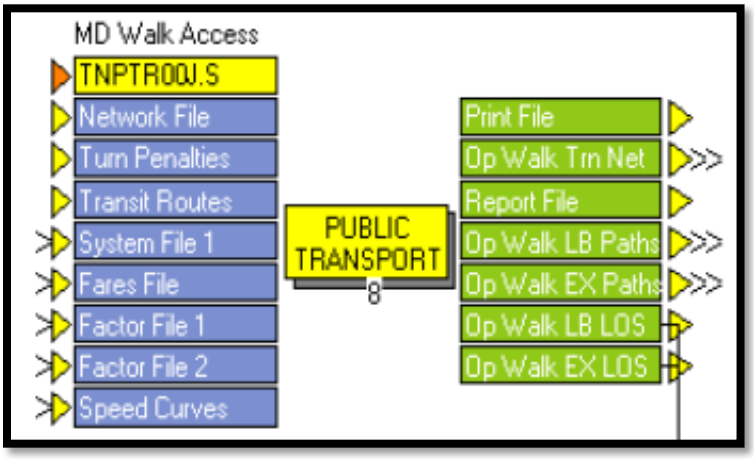
```
; Script for program MATRIX in file "C:\FSUTMS\DISTRICT2\ALACHUA\TNMAT00C.S"
;  Do  not  change  filenames  or  add  or  remove  FILEI/FILEO  statements  using  an  editor.  Use
Cube/Application Manager.
RUN PGM=MATRIX PRNFILE="{SCENARIO_DIR}\output\TNMAT00C.PRN" MSG='Build AM PNR Cost Matrix'
FILEI MATI[2] = "{SCENARIO_DIR}\OUTPUT\RHSKIMS.MAT"
FILEI MATI[1] = "{SCENARIO_DIR}\output\AMPCOST.DBF",
PATTERN=IJ:MV, FIELDS=ORZ,DSZ,MA,MEANCOST,MB,STNZONE,MC,CNT
FILEO MATO[1] = "{SCENARIO_DIR}\output\PKPNRCOST.MAT",
 MO=1-4,13,14 NAME=PKPNRCOST,STNZONE,STNTIME,STNDIST,FREQUENCY,TERMTIME

PAR ZONEMSG=100 ZONES={ZONESA}
MW[1]=MI.1.1 ; PNR COST
MW[2]=MI.1.2 ; STNZONE
MW[13]=MI.1.3 ; FREQENCY COUNT
MW[14]=MI.2.TERMINALTIME
MW[10]=MI.2.TIME
MW[11]=MI.2.DISTANCE
jloop
  IF (MW[13]>0)
     MW[2]=MW[2]/MW[13]
     STNZONE=MW[2]
     TIME=MW[10]
     DISTANCE=MW[11]
     MW[3]=TIME, MW[4]=DISTANCE
  ENDIF
endjloop
ENDRUN
```



TNMAT00I.S

```
; Script for program MATRIX in file "C:\FSUTMS\DISTRICT2\ALACHUA\TNMAT00A.S"
;  Do  not  change  filenames  or  add  or  remove  FILEI/FILEO  statements  using  an  editor.  Use
Cube/Application Manager.
RUN PGM=MATRIX PRNFILE="{SCENARIO_DIR}\output\TNMAT00A.PRN" MSG='Compile AM LOS for Mode Choice'
FILEI MATI[4] = "{SCENARIO_DIR}\output\PKPNRCOST.MAT"
FILEO MATO[1] = "{SCENARIO_DIR}\output\PEAK TRN LOS.MAT",
MO=1-5,11-15,21-25,
NAME=PKWKTIMELB,PKWTTIMELB,PKIVTIMELB,PKPKCOSTLB,PKOPCOSTLB,
PKWKTIMEEX,PKWTTIMEEX,PKIVTIMEEX,PKPKCOSTEX,PKOPCOSTEX,
PKWKTIMEBA,PKWTTIMEBA,PKIVTIMEBA,PKPKCOSTBA,PKOPCOSTBA dec=15*d
FILEI MATI[3] = "{SCENARIO_DIR}\output\AUTOAM.MAT"
FILEI MATI[2] = "{SCENARIO_DIR}\output\WALKPREMAM.MAT"
FILEI MATI[1] = "{SCENARIO_DIR}\output\WALKAM.MAT"

par zonemsg=100

JLOOP

; FIRST PROCESS THE WALK TO LOCAL BUS

IF (MI.1.TIME6M=0&MI.1.TIME8M=0)
   MW[001]=mi.1.ovtt+mi.4.termtime
   MW[002]=mi.1.iwait+mi.1.xwait
   MW[003]=mi.1.ivtt
   MW[004]=0 ; no parking cost for walk modes
   MW[005]=mi.1.fare
ELSE
   MW[013]=999999
ENDIF

; NEXT PROCESS WALK TO EXPRESS SERVICE

IF (MI.1.TIME6M>0|MI.1.TIME8M>0)
   MW[011]=mi.2.ovtt+mi.4.termtime
   MW[012]=mi.2.iwait+mi.2.xwait
   MW[013]=mi.2.ivtt
   MW[014]=0 ; no parking cost for walk modes
   MW[015]=mi.1.fare
ELSE
   MW[013]=999999
ENDIF

; NEXT PROCESS DRIVE TO BEST AVAILABLE
IF (mi.3.ivtt>0)
   MW[021]=mi.3.ovtt+mi.4.termtime
   MW[022]=mi.3.iwait+mi.3.xwait
   MW[023]=mi.3.ivtt+mi.4.stntime
   MW[024]=mi.4.pkpnrcost+(mi.4.stndist*{hwyopcost})
   MW[025]=mi.3.fare
ELSE
   MW[023]=999999
ENDIF
```

```
ENDJLOOP
ENDRUN
```

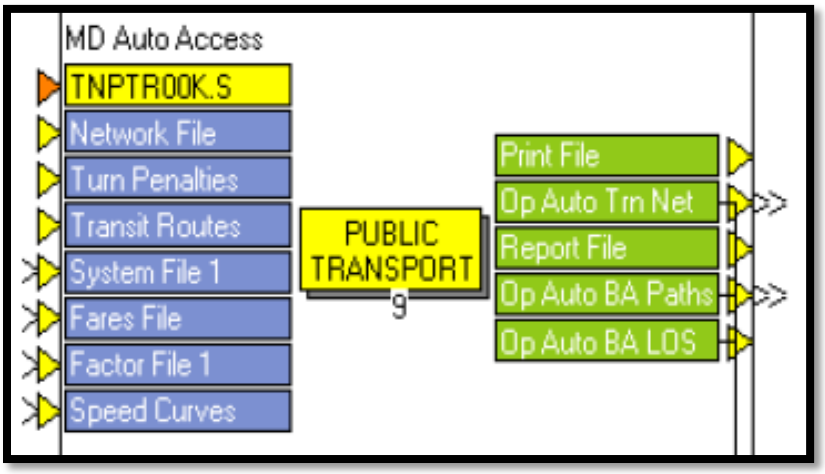


TNPTR00J.S

```
; Script for program PUBLIC TRANSPORT in file "C:\FSUTMS\DISTRICT2\ALACHUA\TNPTR00A.S"
; Do not change filenames or add or remove FILEI/FILEO statements using an editor. Use
Cube/Application Manager.
RUN PGM=PUBLIC TRANSPORT PRNFILE="{SCENARIO_DIR}\output\TNPTR00B.PRN" MSG='MD Walk Access'
FILEI NETI = "{SCENARIO_DIR}\output\UNLOADED.NET"
FILEI LOOKUPI[1] = "{CATALOG_DIR}\Parameters\SPDCRV.CSV"
FILEO MATO[2] = "{SCENARIO_DIR}\output\WALKPREMMD.MAT",
 MO=1-11, NAME=TIME1M,TIME2M,TIME3M,TIME4M,TIME6M,TIME8M,IWAIT,XWAIT,IVTT,OVTT,FARE
FILEO MATO[1] = "{SCENARIO_DIR}\output\WALKMD.MAT",
 MO=1-11, NAME=TIME1M,TIME2M,TIME3M,TIME4M,TIME6M,TIME8M,IWAIT, XWAIT,IVTT,OVTT,FARE
FILEI FACTORI[2] = "{CATALOG_DIR}\Parameters\ALACHUAWKPREM.FAC"
FILEI FACTORI[1] = "{CATALOG_DIR}\Parameters\ALACHUAWLB.FAC"
FILEI FAREI = "{CATALOG_DIR}\parameters\ALACHUA.FAR"
FILEO ROUTEO[2] = "{SCENARIO_DIR}\output\WALKPREMMD.RTE",
 REPORTI=1-{zonesa}, REPORTJ={cbdzone}
FILEO ROUTEO[1] = "{SCENARIO_DIR}\output\WALKLBMD.RTE",
 REPORTI=1-{zonesa}, REPORTJ={cbdzone}
FILEI SYSTEMI = "{CATALOG_DIR}\Parameters\ALACHUA.PTS"
FILEO REPORTO = "{SCENARIO_DIR}\output\TNPTR00A.PRN"
FILEO NETO = "{SCENARIO_DIR}\output\TNETWALKMD.NET"
FILEI LINEI[1] = "{SCENARIO_DIR}\input\troute20{YEAR}.lin"
FILEI TURNPENI = "{SCENARIO_DIR}\input\TCARDS.PEN"
PARAMETERS TRANTIME=(LI.TIME*1.5), ;GENERIC TRANSIT VS. AUTO RUN TIME
           TRANTIME[4]=LW.LBTIME,    ; MODE SPECIFIC TRANSIT VS. AUTO RUN TIME
           TRANTIME[6]=LW.EBTIME,    ; CREATED THROUGH A LOOKUP FUNCTION
           TRANTIME[8]=LW.RLTIME,
           FARE=F, USERCLASSES=1-2,  HDWAYPERIOD=2   MAPSCALE={UNITS}
REPORT LINES=T

;PROCESS PHASE=NODEREAD
; loops over all nodes computes node based scalar and array variables (Optional)
;ENDPROCESS

PROCESS PHASE=LINKREAD
   LW.WALKTIME=60*LI.DISTANCE/{walkspeed}
   IF (LI.TIME>0) LW.SPEED=60*LI.DISTANCE/LI.TIME
   LOOKUP, NAME=CURVES, LOOKUP[1]=1, RESULT=2, LOOKUP[2]=1, RESULT=3, LOOKUP[3]=1, RESULT=4,
           INTERPOLATE=Y, LOOKUPI=1
   IF (LI.FTYPE=10-19,80-99) ; FREE FLOW CONDITIONS FOR ALL
       LW.LBCURVE=1,LW.EBCURVE=1,LW.RLCURVE=1
   ELSEIF (LI.FTYPE=20-79&LI.ATYPE=10-19) ; BUSSES HITTING RESISTANCE, RAIL NO CONFLICTS (GRADE
SEP)
       LW.LBCURVE=2,LW.EBCURVE=2,LW.RLCURVE=1
   ELSEIF (LI.FTYPE=20-79&LI.ATYPE=20-29) ; BUSSES HITTING RESISTANCE(LB MORE), RAIL NO CONFLICTS
       LW.LBCURVE=3,LW.EBCURVE=2,LW.RLCURVE=1
   ELSEIF (LI.FTYPE=20-79&LI.ATYPE=30-39) ; BUSSES HITTING RESISTANCE(LB MORE), RAIL NO CONFLICTS
       LW.LBCURVE=3,LW.EBCURVE=2,LW.RLCURVE=1
```

```
   ELSEIF (LI.FTYPE=20-79&LI.ATYPE=40-49) ; BUSSES HITTING RESISTANCE(LB MORE), RAIL NO CONFLICTS
       LW.LBCURVE=2,LW.EBCURVE=2,LW.RLCURVE=1
   ELSEIF (LI.FTYPE=20-79&LI.ATYPE=50-59) ; BUSSES HITTING RESISTANCE(LB MORE), RAIL NO CONFLICTS
       LW.LBCURVE=2,LW.EBCURVE=1,LW.RLCURVE=1
   ENDIF
       LW.LBSPEED=CURVES(LW.LBCURVE,LW.SPEED)
       LW.EBSPEED=CURVES(LW.EBCURVE,LW.SPEED)
       LW.RLSPEED=CURVES(LW.RLCURVE,LW.SPEED)
       LW.LBTIME=60*LI.DISTANCE/LW.LBSPEED
       LW.EBTIME=60*LI.DISTANCE/LW.EBSPEED
       LW.RLTIME=60*LI.DISTANCE/LW.RLSPEED
       TRANTIME[4]=LW.LBTIME
       TRANTIME[6]=LW.EBTIME
       TRANTIME[8]=LW.RLTIME
   ENDPROCESS

PROCESS PHASE=DATAPREP
; WALK ACCESS
  GENERATE, COST=(LW.WALKTIME),MAXCOST=103*24.0,LIST=T,NTLEGMODE = 1,
            DIRECTION=1, FROMNODE=1-{zonesa}, TONODE=1000-99999
; WALK EGRESS
  GENERATE, COST=(LW.WALKTIME),MAXCOST=103*24.0,LIST=T,NTLEGMODE=101,
            DIRECTION=2, FROMNODE=1-{zonesa}, TONODE=1000-99999
; WALK CONNECTORS
  GENERATE, COST=(LW.WALKTIME),MAXCOST=103*12,LIST=T,NTLEGMODE = 3,DIRECTION=3,
            FROMNODE=1000-99999, TONODE=1000-99999
ENDPROCESS

;PROCESS PHASE=MATI
; manipulates input and work matrices prior to processing each Origin zone, I (Optional)
;ENDPROCESS

;PROCESS PHASE=SELECTIJ
; allows finer selection of zone pairs, IJ, for Route Evaluation, and the setting
; or revising of trips for Loading (Optional)
;ENDPROCESS

PROCESS PHASE=SKIMIJ
  MW[1]=TIMEA(0,1,101)
  MW[2]=TIMEA(0,2,102)
  MW[3]=TIMEA(0,3)
  MW[4]=TIMEA(0,4)
  MW[5]=TIMEA(0,6)
  MW[6]=TIMEA(0,8)
  MW[7]=IWAITA(0)
  MW[8]=XWAITA(0)
  MW[9]=TIMEA(0,TMODES)
  MW[10]=TIMEA(0,NTMODES)
  MW[11]=FAREA(0,ALLMODES)

;VARIOUS THINGS THAT CAN BE SKIMMED
/*
  COMPCOST(RouteSet)        Skims Composite Costs
  ValOfChoice(RouteSet)     Skims Value of Choice
  IWAITA(RouteSet)          Skims Initial Wait Times Actual
  XWAITA(RouteSet)          Skims Transfer Wait Times Actual
  IWAITP(RouteSet)          Skims Initial Wait Times Perceived
  XWAITP(RouteSet)          Skims Initial Transfer Times Perceived
  TIMEA(RouteSet, Mode)     Skims Travel Time Actual
  TIMEP(RouteSet, Mode)     Skims Travel Time Perceived

  XFERPENA(RouteSet, Mode)  Skims Transfer Penalty Actual

  XFERPENP(RouteSet, Mode)  Skims Transfer Penalty Actual
  DIST(RouteSet, Mode)      Skims Distance
  BRDINGS(RouteSet, Mode)   Skims Number of Boardings (xfers+1)
  BESTJRNY                  Skims Best Journey Times
  FAREA(RouteSet, Mode)     Skims Fares in Monetary units
  FAREP(RouteSet, Mode)     Skims Fares in Generalized Time units
*/
ENDPROCESS
```

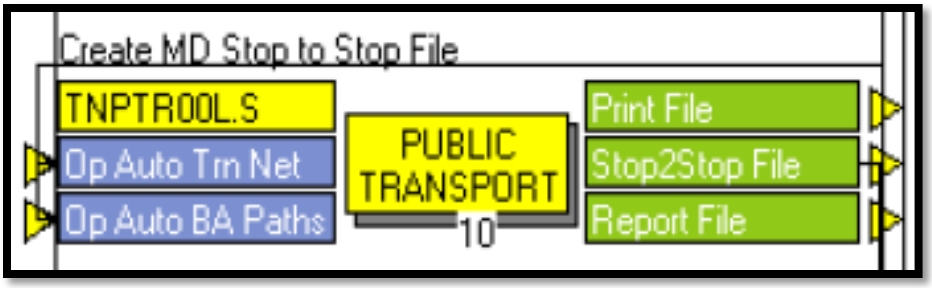
```
ENDRUN
```



TNPTR00K.S

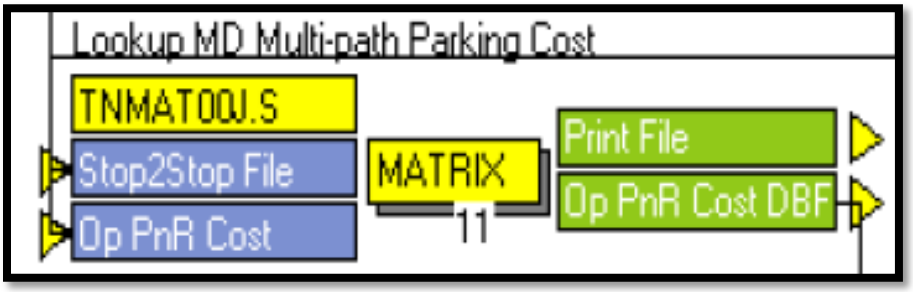
```
; Script for program PUBLIC TRANSPORT in file "C:\FSUTMS\DISTRICT2\ALACHUA\TNPTR00B.S"
; Do not change filenames or add or remove FILEI/FILEO statements using an editor. Use
Cube/Application Manager.
RUN PGM=PUBLIC TRANSPORT PRNFILE="{SCENARIO_DIR}\output\TNPTR00H.PRN" MSG='MD Auto Access'
FILEI NETI = "{SCENARIO_DIR}\output\UNLOADED.NET"
FILEI LOOKUPI[1] = "{CATALOG_DIR}\Parameters\SPDCRV.CSV"
FILEO REPORTO = "{SCENARIO_DIR}\output\TNPTR00E.PRN"
FILEO MATO[1] = "{SCENARIO_DIR}\output\AUTOMD.MAT",
 MO=1-11, NAME=TIME1M,TIME2M,TIME3M,TIME4M,TIME6M,TIME8M,IWAIT,XWAIT,IVTT,OVTT,FARE
FILEI FACTORI[1] = "{CATALOG_DIR}\Parameters\ALACHUAPNR.FAC"
FILEI FAREI = "{CATALOG_DIR}\parameters\ALACHUA.FAR"
FILEO ROUTEO[1] = "{SCENARIO_DIR}\output\AUTOALLMD.RTE",
 REPORTI=1-{zonesa}, REPORTJ={cbdzone}
FILEI SYSTEMI = "{CATALOG_DIR}\Parameters\ALACHUA.PTS"
FILEI LINEI[1] = "{SCENARIO_DIR}\input\troute20{YEAR}.lin"
FILEI TURNPENI = "{SCENARIO_DIR}\input\TCARDS.PEN"
FILEO NETO = "{SCENARIO_DIR}\output\TNETAUTOMD.NET"
PARAMETERS TRANTIME=(LI.TIME*1.5), ;GENERIC TRANSIT VS. AUTO RUN TIME
           TRANTIME[4]=LW.LBTIME,    ; MODE SPECIFIC TRANSIT VS. AUTO RUN TIME
           TRANTIME[6]=LW.EBTIME,    ; CREATED THROUGH A LOOKUP FUNCTION
           TRANTIME[8]=LW.RLTIME,
           FARE=F, USERCLASSES=1,    MAPSCALE={UNITS}
           HDWAYPERIOD=2
REPORT LINES=T

;PROCESS PHASE=NODEREAD
; loops over all nodes computes node based scalar and array variables (Optional)
;ENDPROCESS

PROCESS PHASE=LINKREAD
   LW.WALKTIME=60*LI.DISTANCE/{walkspeed}
   IF (LI.TIME>0) LW.SPEED=60*LI.DISTANCE/LI.TIME
   LOOKUP, NAME=CURVES, LOOKUP[1]=1, RESULT=2, LOOKUP[2]=1, RESULT=3, LOOKUP[3]=1, RESULT=4,
           INTERPOLATE=Y, LOOKUPI=1
   IF (LI.FTYPE=10-19,80-99) ; FREE FLOW CONDITIONS FOR ALL
       LW.LBCURVE=1,LW.EBCURVE=1,LW.RLCURVE=1
   ELSEIF (LI.FTYPE=20-79&LI.ATYPE=10-19) ; BUSSES HITTING RESISTANCE, RAIL NO CONFLICTS (GRADE
SEP)
       LW.LBCURVE=2,LW.EBCURVE=2,LW.RLCURVE=1
   ELSEIF (LI.FTYPE=20-79&LI.ATYPE=20-29) ; BUSSES HITTING RESISTANCE(LB MORE), RAIL NO CONFLICTS
       LW.LBCURVE=3,LW.EBCURVE=2,LW.RLCURVE=1
   ELSEIF (LI.FTYPE=20-79&LI.ATYPE=30-39) ; BUSSES HITTING RESISTANCE(LB MORE), RAIL NO CONFLICTS
       LW.LBCURVE=3,LW.EBCURVE=2,LW.RLCURVE=1
   ELSEIF (LI.FTYPE=20-79&LI.ATYPE=40-49) ; BUSSES HITTING RESISTANCE(LB MORE), RAIL NO CONFLICTS
       LW.LBCURVE=2,LW.EBCURVE=2,LW.RLCURVE=1
   ELSEIF (LI.FTYPE=20-79&LI.ATYPE=50-59) ; BUSSES HITTING RESISTANCE(LB MORE), RAIL NO CONFLICTS
```

```
        LW.LBCURVE=2,LW.EBCURVE=1,LW.RLCURVE=1
    ENDIF
        LW.LBSPEED=CURVES(LW.LBCURVE,LW.SPEED)
        LW.EBSPEED=CURVES(LW.EBCURVE,LW.SPEED)
        LW.RLSPEED=CURVES(LW.RLCURVE,LW.SPEED)
        LW.LBTIME=60*LI.DISTANCE/LW.LBSPEED
        LW.EBTIME=60*LI.DISTANCE/LW.EBSPEED
        LW.RLTIME=60*LI.DISTANCE/LW.RLSPEED
        TRANTIME[4]=LW.LBTIME
        TRANTIME[6]=LW.EBTIME
        TRANTIME[8]=LW.RLTIME
    ENDPROCESS

PROCESS PHASE=DATAPREP
; AUTO ACCESS
READ,
 FILE = "{SCENARIO_DIR}\OUTPUT\MD_STATDATA.CSV"
; WALK EGRESS
  GENERATE, COST=(LW.WALKTIME),MAXCOST=103*24.0,LIST=T,NTLEGMODE=101,
            DIRECTION=2, FROMNODE=1-{zonesa}, TONODE=1000-99999
; WALK CONNECTORS
  GENERATE, COST=(LW.WALKTIME),MAXCOST=103*12,LIST=T,NTLEGMODE = 3,DIRECTION=3,
            FROMNODE=1000-99999, TONODE=1000-99999
ENDPROCESS

;PROCESS PHASE=MATI
; manipulates input and work matrices prior to processing each Origin zone, I (Optional)
;ENDPROCESS
;PROCESS PHASE=SELECTIJ
; allows finer selection of zone pairs, IJ, for Route Evaluation, and the setting
; or revising of trips for Loading (Optional)
;ENDPROCESS

PROCESS PHASE=SKIMIJ
  MW[1]=TIMEA(0,1,101)
  MW[2]=TIMEA(0,2,102)
  MW[3]=TIMEA(0,3)
  MW[4]=TIMEA(0,4)
  MW[5]=TIMEA(0,6)
  MW[6]=TIMEA(0,8)
  MW[7]=IWAITA(0)
  MW[8]=XWAITA(0)
  MW[9]=TIMEA(0,TMODES)
  MW[10]=TIMEA(0,NTMODES)
  MW[11]=FAREA(0,ALLMODES)

;VARIOUS THINGS THAT CAN BE SKIMMED
/*
  COMPCOST(RouteSet)        Skims Composite Costs
  ValOfChoice(RouteSet)     Skims Value of Choice
  IWAITA(RouteSet)          Skims Initial Wait Times Actual
  XWAITA(RouteSet)          Skims Transfer Wait Times Actual
  IWAITP(RouteSet)          Skims Initial Wait Times Perceived
  XWAITP(RouteSet)          Skims Initial Transfer Times Perceived
  TIMEA(RouteSet, Mode)     Skims Travel Time Actual
  TIMEP(RouteSet, Mode)     Skims Travel Time Perceived
  XFERPENA(RouteSet, Mode)  Skims Transfer Penalty Actual
  XFERPENP(RouteSet, Mode)  Skims Transfer Penalty Actual
  DIST(RouteSet, Mode)      Skims Distance
  BRDINGS(RouteSet, Mode)   Skims Number of Boardings (xfers+1)
  BESTJRNY                  Skims Best Journey Times
  FAREA(RouteSet, Mode)     Skims Fares in Monetary units
  FAREP(RouteSet, Mode)     Skims Fares in Generalized Time units
*/
ENDPROCESS
;PROCESS PHASE=MATO
; allows processing of work matrices prior to them being written to the MATO files
; at the end of each Origin zone (Optional)
;ENDPROCESS
ENDRUN
```



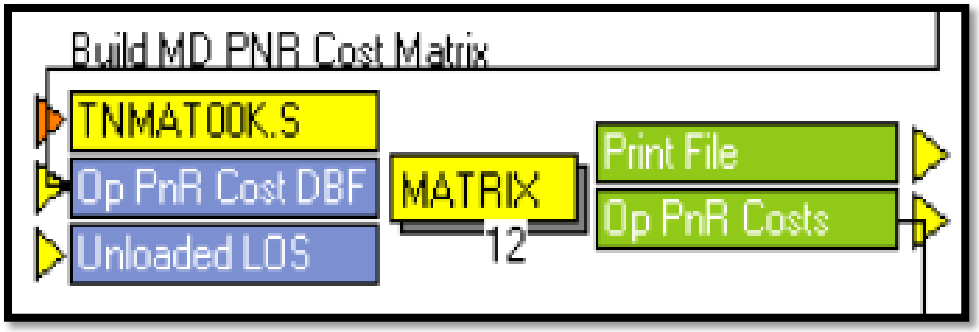
TNPTR00L.S

```
; Script for program PUBLIC TRANSPORT in file "C:\FSUTMS\DISTRICT2\ALACHUA\TNPTR00F.S"
; Do not change filenames or add or remove FILEI/FILEO statements using an editor. Use
Cube/Application Manager.
RUN PGM=PUBLIC TRANSPORT PRNFILE="{SCENARIO_DIR}\output\TNPTR00K.PRN" MSG='Create MD Stop to Stop
File'
FILEI ROUTEI[1] = "{SCENARIO_DIR}\output\AUTOALLMD.RTE"
FILEI NETI = "{SCENARIO_DIR}\output\TNETAUTOMD.NET"
FILEO REPORTO = "{SCENARIO_DIR}\output\TNPTR00L.PRN"
FILEO STOP2STOPO = "{SCENARIO_DIR}\output\TNPTR00C.DBF",
 ACCUMULATE=FIRSTLAST, NODES=1-99999
PARAMETERS HDWAYPERIOD=2,
           TRIPSIJ[1]=100,
           NOROUTEERRS=999999999
ENDRUN
```



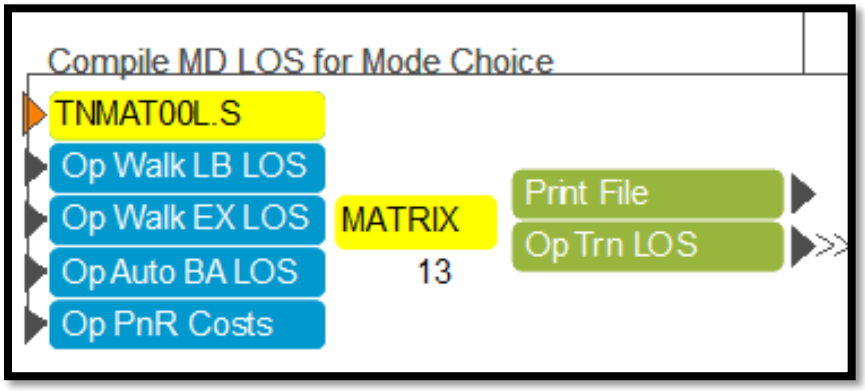
TNMAT00J.S

```
; Script for program MATRIX in file "C:\FSUTMS\DISTRICT2\ALACHUA\TNMAT00D.S"
; Do not change filenames or add or remove FILEI/FILEO statements using an editor. Use
Cube/Application Manager.
RUN PGM=MATRIX PRNFILE="{SCENARIO_DIR}\output\TNMAT00D.PRN" MSG='Lookup MD Multi-path Parking Cost'
FILEI RECI = "{SCENARIO_DIR}\output\TNPTR00C.DBF"
FILEO RECO[1] = "{SCENARIO_DIR}\output\MDPCOST.DBF",
 FIELDS=ORZ,DSZ,MA,MEANCOST,MB,STNZONE,MC,CNT
FILEI LOOKUPI[1] = "{SCENARIO_DIR}\output\OPPNRCOST.CSV"
LOOKUP, NAME=STATIONS, LOOKUP[1]=1, RESULT=2, LOOKUP[2]=1, RESULT=3, INTERPOLATE=F, FAIL[1]=0,
FAIL[2]=0, LOOKUPI=1
  RO.ORZ=RI.I
  RO.DSZ=RI.J
  statnode=ri.fromnode
  RO.MA=1
  RO.MB=2
  RO.MC=3
  RO.CNT=1
  PCOST=STATIONS(1,statnode)
  RO.STNZONE=STATIONS(2,STATNODE)
  meancost=PCOST*ri.vol/100

WRITE RECO=1
ENDRUN
```



TNMAT00K.S

```
; Script for program MATRIX in file "C:\FSUTMS\DISTRICT2\ALACHUA\TNMAT00F.S"
; Do not change filenames or add or remove FILEI/FILEO statements using an editor. Use
Cube/Application Manager.
RUN PGM=MATRIX PRNFILE="{SCENARIO_DIR}\output\TNMAT00F.PRN" MSG='Build MD PNR Cost Matrix'
FILEI MATI[2] = "{SCENARIO_DIR}\Output\FHSKIMS.{ALT}{YEAR}.MAT"
FILEI MATI[1] = "{SCENARIO_DIR}\output\MDPCOST.DBF",
PATTERN=IJ:MV, FIELDS=ORZ,DSZ,MA,MEANCOST,MB,STNZONE,MC,CNT
FILEO MATO[1] = "{SCENARIO_DIR}\output\OPPNRCOST.MAT",
 MO=1-4,13,14 NAME=PKPNRCOST,STNZONE,STNTIME,STNDIST,FREQUENCY,TERMTIME

PAR ZONEMSG=100 ZONES={ZONESA}
MW[1]=MI.1.1 ; PNR COST
MW[2]=MI.1.2 ; STNZONE
MW[13]=MI.1.3 ; FREQENCY COUNT
MW[14]=MI.2.TERMINALTIME
MW[10]=MI.2.TIME
MW[11]=MI.2.DISTANCE
jloop
  IF (MW[13]>0)
     MW[2]=MW[2]/MW[13]
     STNZONE=MW[2]
     TIME=MW[10]
     DISTANCE=MW[11]
     MW[3]=TIME, MW[4]=DISTANCE
  ENDIF
endjloop
PAR ZONES={ZONESA}
MW[1]=MI.1.1
ENDRUN
```

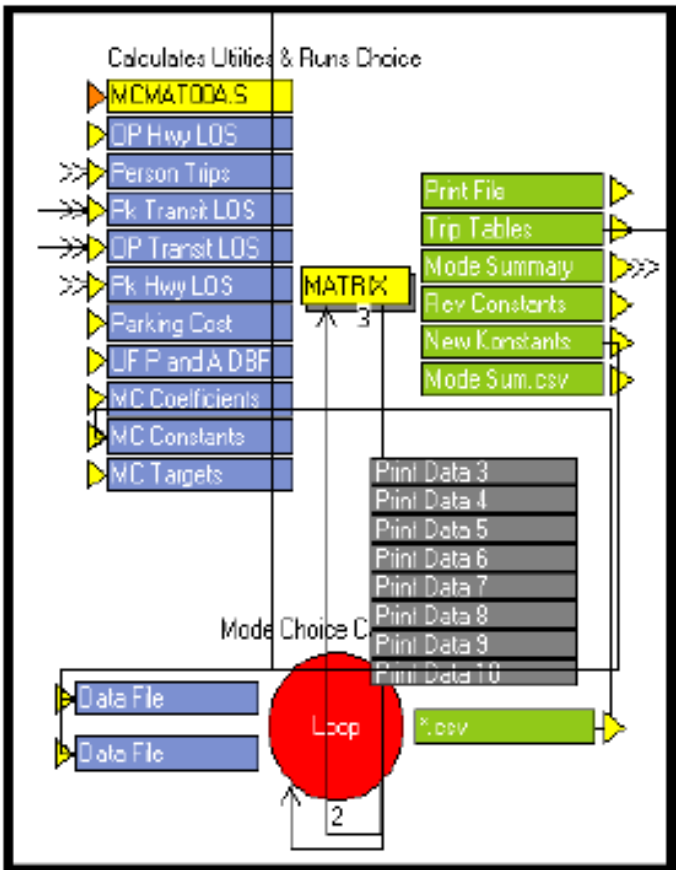


TNMAT00L.S

```
; Script for program MATRIX in file "C:\FSUTMS\DISTRICT2\ALACHUA\TNMAT00B.S"
; Do not change filenames or add or remove FILEI/FILEO statements using an editor. Use
Cube/Application Manager.
RUN PGM=MATRIX PRNFILE="{SCENARIO_DIR}\output\TNMAT00B.PRN" MSG='Compile MD LOS for Mode Choice'
```

```
FILEI MATI[4] = "{SCENARIO_DIR}\output\OPPNRCOST.MAT"
FILEI MATI[3] = "{SCENARIO_DIR}\output\AUTOMD.MAT"
FILEI MATI[2] = "{SCENARIO_DIR}\output\WALKPREMMD.MAT"
FILEI MATI[1] = "{SCENARIO_DIR}\output\WALKMD.MAT"
FILEO MATO[1] = "{SCENARIO_DIR}\output\OP TRN LOS.MAT",
MO=1-5,11-15,21-25,
NAME=OPWKTIMELB,OPWTTIMELB,OPIVTIMELB,OPPKCOSTLB,OPOPCOSTLB,
OPWKTIMEEX,OPWTTIMEEX,OPIVTIMEEX,OPPKCOSTEX,OPOPCOSTEX,
OPWKTIMEBA,OPWTTIMEBA,OPIVTIMEBA,OPPKCOSTBA,OPOPCOSTBA DEC=15*D

par zonemsg=100

JLOOP

IF (MI.1.TIME6M=0&MI.1.TIME8M=0)
   MW[001]=mi.1.ovtt+mi.4.termtime
   MW[002]=mi.1.iwait+mi.1.xwait
   MW[003]=mi.1.ivtt
   MW[004]=0 ; no parking cost for walk modes
   MW[005]=mi.1.fare
ELSE
   MW[003]=999999
ENDIF

; NEXT PROCESS WALK TO EXPRESS SERVICE

IF (MI.1.TIME6M>0|MI.1.TIME8M>0)
   MW[011]=mi.2.ovtt+mi.4.termtime
   MW[012]=mi.2.iwait+mi.2.xwait
   MW[013]=mi.2.ivtt
   MW[014]=0 ; no parking cost for walk modes
   MW[015]=mi.1.fare
ELSE
   MW[013]=999999
ENDIF

; NEXT PROCESS DRIVE TO BEST AVAILABLE
IF (mi.3.ivtt>0)
   MW[021]=mi.3.ovtt+mi.4.termtime
   MW[022]=mi.3.iwait+mi.3.xwait
   MW[023]=mi.3.ivtt+mi.4.stntime
   MW[024]=mi.4.pkpnrcost+(mi.4.stndist*{hwyopcost})
   MW[025]=mi.3.fare
ELSE
   MW[023]=999999
ENDIF

ENDJLOOP
ENDRUN
```

Mode Choice Step



MCMAT00A.S

```
; Do not change filenames or add or remove FILEI/FILEO statements using an editor. Use
Cube/Application Manager.
RUN PGM=MATRIX PRNFILE="{SCENARIO_DIR}\output\MCMAT00A.PRN" MSG='Calculates Utiities & Runs Choice'
FILEI MATI[1] = "{SCENARIO_DIR}\output\FHSKIMS.{ALT}{YEAR}.MAT"
FILEI ZDATI[3] = "{SCENARIO_DIR}\output\UFPANDA.DBF"
FILEI MATI[5] = "{SCENARIO_DIR}\output\RHSKIMS.MAT"
FILEI MATI[4] = "{SCENARIO_DIR}\output\OP TRN LOS.MAT"
FILEI MATI[3] = "{SCENARIO_DIR}\output\PEAK TRN LOS.MAT"
FILEI MATI[2] = "{SCENARIO_DIR}\output\PTRIPS.MAT"
FILEO PRINTO[4] = "{SCENARIO_DIR}\output\MODE SUM.CSV"
FILEI ZDATI[1] = "{SCENARIO_DIR}\input\ZoneData{YEAR}.DBF",
Z=TAZ_20{year}
FILEI LOOKUPI[2] = "{SCENARIO_DIR}\output\MCLOO00C.CSV"
FILEO PRINTO[3] = "{SCENARIO_DIR}\output\NEWK.CSV"
FILEI LOOKUPI[3] = "{CATALOG_DIR}\parameters\MC_TARGETS.CSV"
FILEO PRINTO[2] = "{SCENARIO_DIR}\output\REV_MODE_CONST.CSV"
FILEO PRINTO[1] = "{SCENARIO_DIR}\output\MODE SUMMARY.PRN"
FILEO MATO[2] = "{SCENARIO_DIR}\output\MODEOUT.MAT",
mo=151-158,161-168,171-178,181-188,191-193,
 name=HBWDA,HBWCP,HBWCX,HBWWB,HBWWX,HBWBA,HBWWK,HBWBK,
      HBODA,HBOCP,HBOCX,HBOWB,HBOWX,HBOBA,HBOWK,HBOBK,
      NHBDA,NHBCP,NHBCX,NHBWB,NHBWX,NHBBA,NHBWK,NHBBK,
      HBUDA,HBUCP,HBUCX,HBUWB,HBUWX,HBUBA,HBUWK,HBUBK,
      HDORMUWB,HDORMUWK,HDORMUBK, DEC=24*S
FILEI LOOKUPI[1] = "{CATALOG_DIR}\parameters\MC_COEFFICIENTS.CSV"
par zonemsg=100

; THE JOB OF THIS SCRIPT IS TO TURN THE COMPONENTS OF UTILITY FOR EACH MODE IN THE MODE CHOICE
```

```
; MODEL INTO A COMPOSITE UTILITY.  BECAUSE THE MODEL IS NESTED, WITH NESTING COEFFICIENTS APPLIED
; IN THE MODE CHOICE MODE, THE INPUT UTILITIES SHOULD BE DIVIDED BY THE PRODUCT OF THE NESTING
COEFFICIENTS.

; MARKET SEGMENTS ARE:
; 0 CAR HOUSEHOLDS
; 1 OR MORE CAR HOUSEHOLDS
; UNIVERSITY STUDENTS

; TRIP PURPOSES ARE:
; 1 HBW (AM PEAK LOS MATRICES)
; 2 HBO (MD OFF-PEAK LOS)
; 3 NHB (MD OFF-PEAK LOS)
; 4 HBU (MD OFF-PEAK LOS)
; 5 HDORMU (choice set: walk, bike, MD walk-local BUS)
  MW[1]=MI.2.HBW
  MW[2]=MI.2.HBSH+MI.2.HBSR+MI.2.HBO
  MW[3]=MI.2.NHB
  MW[4]=MI.2.HBU
  MW[5]=MI.2.HDORMU

; THE AUTO DIVISOR IS NESTCMOTOR*NESTCAUTO
  NESTMOTOR={NESTCMOTOR}*{NESTCAUTO}
; THE TRANSIT DIVISOR IS NESTCMOTOR*NESTCTRANSIT
  NESTTRANSIT={NESTCMOTOR}*{NESTCTRANSIT}
  NESTNONMOTOR={NESTCNONMOTOR}

;Coefficients
lookup, name=coefficients,
lookup[1]=1, result=2, lookup[2]=1, result=3, lookup[3]=1, result=4, lookup[4]=1, result=5,
interpolate=n, LIST=Y, lookupi=1
; civt-IN VEHICLE TIME COEFFICIENT
HBWCIVT=COEFFICIENTS(1,1),          HBOCIVT=COEFFICIENTS(2,1),          NHBCIVT=COEFFICIENTS(3,1),
UNICIVT=COEFFICIENTS(4,1)
; covt-OUT OF VEHICLE TIME COEFFICIENT
HBWCOVT=COEFFICIENTS(1,2),          HBOCOVT=COEFFICIENTS(2,2),          NHBCOVT=COEFFICIENTS(3,2),
UNICOVT=COEFFICIENTS(4,2)
; ccst-COST COEFFICIENT (cents)
HBWCCST=COEFFICIENTS(1,3),          HBOCCST=COEFFICIENTS(2,3),          NHBCCST=COEFFICIENTS(3,3),
UNICCST=COEFFICIENTS(4,3)
; cwt-WALK ONLY COEFFICIENT
HBWCWT=COEFFICIENTS(1,4),           HBOCWT=COEFFICIENTS(2,4),           NHBCWT=COEFFICIENTS(3,4),
UNICWT=COEFFICIENTS(4,4)
; cbt-BIKE ONLY COEFFICIENT
HBWCBT=COEFFICIENTS(1,5),           HBOCBT=COEFFICIENTS(2,5),           NHBCBT=COEFFICIENTS(3,5),
UNICBT=COEFFICIENTS(4,5)
; pti-Walk to transit PEV i
HBWPTI=COEFFICIENTS(1,6),           HBOPTI=COEFFICIENTS(2,6),           NHBPTI=COEFFICIENTS(3,6),
UNIPTI=COEFFICIENTS(4,6)
; pwi-Walk PEV i
HBWPWI=COEFFICIENTS(1,7),           HBOPWI=COEFFICIENTS(2,7),           NHBPWI=COEFFICIENTS(3,7),
UNIPWI=COEFFICIENTS(4,7)
; pwi-Walk PEV J
HBWPWJ=COEFFICIENTS(1,8),           HBOPWJ=COEFFICIENTS(2,8),           NHBPWJ=COEFFICIENTS(3,8),
UNIPWJ=COEFFICIENTS(4,8)
; pbi-BIKE PEV i
HBWPBI=COEFFICIENTS(1,9),           HBOPBI=COEFFICIENTS(2,9),           NHBPBI=COEFFICIENTS(3,9),
UNIPBI=COEFFICIENTS(4,9)
; pbi-BIKE PEV J
HBWPBJ=COEFFICIENTS(1,10),          HBOPBJ=COEFFICIENTS(2,10),          NHBPBJ=COEFFICIENTS(3,10),
UNIPBJ=COEFFICIENTS(4,10)

;Constants
;3*HBW,HBO,NHB,Constants for 0 car, 1+car and student, rows=mode=da,cp,cx,wl,wx,ab,wk,bk
lookup, name=constants,
lookup[1]=1, result=2, lookup[2]=1, result=3, lookup[3]=1, result=4,
lookup[4]=1, result=5, lookup[5]=1, result=6, lookup[6]=1, result=7,
lookup[7]=1, result=8,lookup[8]=1, result=9,lookup[9]=1, result=10,
interpolate=n, , LIST=Y, lookupi=2
;K=CONSTANT, FOLLOWED BY TRIP PURPOSE FOLLOWED BY MODE
;  DRIVE ALONE
```

```
K1_NC_DA=CONSTANTS(1,1), K2_NC_DA=CONSTANTS(4,1), K3_NC_DA=CONSTANTS(7,1), K4_NC_DA=CONSTANTS(8,1)
K1_WC_DA=CONSTANTS(2,1), K2_WC_DA=CONSTANTS(5,1)                          , K5_NC_DA=CONSTANTS(9,1)
K1_ST_DA=CONSTANTS(3,1), K2_ST_DA=CONSTANTS(6,1)
;  2  CARPOOL
K1_NC_CP=CONSTANTS(1,2), K2_NC_CP=CONSTANTS(4,2), K3_NC_CP=CONSTANTS(7,2), K4_NC_CP=CONSTANTS(8,2)
K1_WC_CP=CONSTANTS(2,2), K2_WC_CP=CONSTANTS(5,2)                          , K5_NC_CP=CONSTANTS(9,2)

K1_ST_CP=CONSTANTS(3,2), K2_ST_CP=CONSTANTS(6,2)
;  3+ CARPOOL
K1_NC_CX=CONSTANTS(1,3), K2_NC_CX=CONSTANTS(4,3), K3_NC_CX=CONSTANTS(7,3), K4_NC_CX=CONSTANTS(8,3)
K1_WC_CX=CONSTANTS(2,3), K2_WC_CX=CONSTANTS(5,3)                          , K5_NC_CX=CONSTANTS(9,3)
K1_ST_CX=CONSTANTS(3,3), K2_ST_CX=CONSTANTS(6,3)
;  WALK TO BUS
K1_NC_WB=CONSTANTS(1,4), K2_NC_WB=CONSTANTS(4,4), K3_NC_WB=CONSTANTS(7,4), K4_NC_WB=CONSTANTS(8,4)
K1_WC_WB=CONSTANTS(2,4), K2_WC_WB=CONSTANTS(5,4)                          , K5_NC_WB=CONSTANTS(9,4)
K1_ST_WB=CONSTANTS(3,4), K2_ST_WB=CONSTANTS(6,4)
;  WALK TO PREMIUM TRANSIT
K1_NC_WX=CONSTANTS(1,5), K2_NC_WX=CONSTANTS(4,5), K3_NC_WX=CONSTANTS(7,5), K4_NC_WX=CONSTANTS(8,5)
K1_WC_WX=CONSTANTS(2,5), K2_WC_WX=CONSTANTS(5,5)                          , K5_NC_WX=CONSTANTS(9,5)
K1_ST_WX=CONSTANTS(3,5), K2_ST_WX=CONSTANTS(6,5)
;  AUTO TO TRANSIT
K1_NC_BA=CONSTANTS(1,6), K2_NC_BA=CONSTANTS(4,6), K3_NC_BA=CONSTANTS(7,6), K4_NC_BA=CONSTANTS(8,6)
K1_WC_BA=CONSTANTS(2,6), K2_WC_BA=CONSTANTS(5,6)                          , K5_NC_BA=CONSTANTS(9,6)
K1_ST_BA=CONSTANTS(3,6), K2_ST_BA=CONSTANTS(6,6)
;  WALK ONLY
K1_NC_WK=CONSTANTS(1,7), K2_NC_WK=CONSTANTS(4,7), K3_NC_WK=CONSTANTS(7,7), K4_NC_WK=CONSTANTS(8,7)
K1_WC_WK=CONSTANTS(2,7), K2_WC_WK=CONSTANTS(5,7)                          , K5_NC_WK=CONSTANTS(9,7)
K1_ST_WK=CONSTANTS(3,7), K2_ST_WK=CONSTANTS(6,7)
;  BIKE ONLY
K1_NC_BK=CONSTANTS(1,8), K2_NC_BK=CONSTANTS(4,8), K3_NC_BK=CONSTANTS(7,8), K4_NC_BK=CONSTANTS(8,8)
K1_WC_BK=CONSTANTS(2,8), K2_WC_BK=CONSTANTS(5,8)                          , K5_NC_BK=CONSTANTS(9,8)
K1_ST_BK=CONSTANTS(3,8), K2_ST_BK=CONSTANTS(6,8)

;TARGETS
;3*HBW,HBO,NHB,Targets for 0 car, 1+car and student, rows=mode=da,cp,cx,wl,wx,ab,wk,bk
lookup, name=targ,
lookup[1]=1, result=2, lookup[2]=1, result=3, lookup[3]=1, result=4,
lookup[4]=1, result=5, lookup[5]=1, result=6, lookup[6]=1, result=7,
lookup[7]=1, result=8,lookup[8]=1, result=9,lookup[9]=1, result=10,
interpolate=n, , LIST=Y, lookupi=3
;t=Target, FOLLOWED BY TRIP PURPOSE FOLLOWED BY MODE
;  DRIVE ALONE
t1_NC_DA=targ(1,1), t2_NC_DA=targ(4,1), t3_NC_DA=targ(7,1), t4_NC_DA=targ(8,1)
t1_WC_DA=targ(2,1), t2_WC_DA=targ(5,1)                    , t5_NC_DA=targ(9,1)
t1_ST_DA=targ(3,1), t2_ST_DA=targ(6,1)
;  2  CARPOOL
t1_NC_CP=targ(1,2), t2_NC_CP=targ(4,2), t3_NC_CP=targ(7,2), t4_NC_CP=targ(8,2)
t1_WC_CP=targ(2,2), t2_WC_CP=targ(5,2)                    , t5_NC_CP=targ(9,2)
t1_ST_CP=targ(3,2), t2_ST_CP=targ(6,2)
;  3+ CARPOOL
t1_NC_CX=targ(1,3), t2_NC_CX=targ(4,3), t3_NC_CX=targ(7,3), t4_NC_CX=targ(8,3)
t1_WC_CX=targ(2,3), t2_WC_CX=targ(5,3)                    , t5_NC_CX=targ(9,3)
t1_ST_CX=targ(3,3), t2_ST_CX=targ(6,3)
;  WALK TO BUS
t1_NC_WB=targ(1,4), t2_NC_WB=targ(4,4), t3_NC_WB=targ(7,4), t4_NC_WB=targ(8,4)
t1_WC_WB=targ(2,4), t2_WC_WB=targ(5,4)                    , t5_NC_WB=targ(9,4)
t1_ST_WB=targ(3,4), t2_ST_WB=targ(6,4)
;  WALK TO PREMIUM TRANSIT
t1_NC_WX=targ(1,5), t2_NC_WX=targ(4,5), t3_NC_WX=targ(7,5), t4_NC_WX=targ(8,5)
t1_WC_WX=targ(2,5), t2_WC_WX=targ(5,5)                    , t5_NC_WX=targ(9,5)
t1_ST_WX=targ(3,5), t2_ST_WX=targ(6,5)
;  AUTO TO TRANSIT
t1_NC_BA=targ(1,6), t2_NC_BA=targ(4,6), t3_NC_BA=targ(7,6), t4_NC_BA=targ(8,6)
t1_WC_BA=targ(2,6), t2_WC_BA=targ(5,6)                    , t5_NC_BA=targ(9,6)
t1_ST_BA=targ(3,6), t2_ST_BA=targ(6,6)
;  WALK ONLY
t1_NC_WK=targ(1,7), t2_NC_WK=targ(4,7), t3_NC_WK=targ(7,7), t4_NC_WK=targ(8,7)
t1_WC_WK=targ(2,7), t2_WC_WK=targ(5,7)                    , t5_NC_WK=targ(9,7)
t1_ST_WK=targ(3,7), t2_ST_WK=targ(6,7)
;  BIKE ONLY
t1_NC_BK=targ(1,8), t2_NC_BK=targ(4,8), t3_NC_BK=targ(7,8), t4_NC_BK=targ(8,8)
```

```
t1_WC_BK=targ(2,8), t2_WC_BK=targ(5,8)                    , t5_NC_BK=targ(9,8)
t1_ST_BK=targ(3,8), t2_ST_BK=targ(6,8)

; COST UNITS
; assume parking costs are in cents, both for auto and PnR lots
; assume fares are in dollars, so multiply by 100.
; assume auto operating costs are in dollars, so multiply by 100.
; Bus fare factor
; Difference of bus fare factors between year 2007 (1.0) and
; future scenarios (1.5) will be reduced to 10% of actual difference amount
; due to significant impact from this bus fare increase since year 2007 ($1.00 to $1.50).
busfarefac=1+(({BUSFAREFAC}-1)*0.10)

    JLOOP
; =====================================================================
; HBW (PEAK) TRIP PURPOSE
; =====================================================================

  ;   PEAK PERIOD DRIVE ALONE ELEMENTS OF UTILITY ARE:
  ;   WALK TIME
      MW[11]=(MI.5.TERMINALTIME)*HBWCOVT
  ;   WAIT TIME
      ;MW[12]=(0)*HBWCOVT
  ;   IVTT
      MW[13]=(MI.5.TIME)*HBWCIVT
  ;   PARKING COST - ONLY AT DESTINATION (J), HALF IN EACH DIRECTION
      MW[14]=0.5 * ZI.1.LONGPARK[J] * HBWCCST
  ;   OTHER COST
      MW[15]=MI.5.DISTANCE * {HWYOPCOST} * 100 * HBWCCST
  ;   COMPOSITE UTILITY
      ;MW[021]=(MW[11]+ MW[13]+MW[14]+MW[15]+K1_NC_DA)/NESTMOTOR
      MW[031]=(MW[11]+ MW[13]+MW[14]+MW[15]+K1_WC_DA)/NESTMOTOR
      MW[041]=(MW[11]+ MW[13]+MW[14]+MW[15]+K1_ST_DA)/NESTMOTOR

  ;   PEAK PERIOD CARPOOL2 ELEMENTS OF UTILITY ARE:
  ;   WALK TIME
      MW[11]=(MI.5.TERMINALTIME)*HBWCOVT
  ;   WAIT TIME
      ;MW[12]=(0)*HBWCOVT
  ;   IVTT
      MW[13]=(MI.5.TIME)*HBWCIVT
  ;   PARKING COST - ONLY AT DESTINATION (J), HALF IN EACH DIRECTION, SHARED BY 2 = 0.25
      MW[14]=0.25 * ZI.1.LONGPARK[J] * HBWCCST
  ;   OTHER COST
      MW[15]= 0.50 * MI.5.DISTANCE * {HWYOPCOST} * 100 * HBWCCST
  ;   COMPOSITE UTILITY
      MW[022]=(MW[11]+ MW[13]+MW[14]+MW[15]+K1_NC_CP)/NESTMOTOR
      MW[032]=(MW[11]+ MW[13]+MW[14]+MW[15]+K1_WC_CP)/NESTMOTOR
      MW[042]=(MW[11]+ MW[13]+MW[14]+MW[15]+K1_ST_CP)/NESTMOTOR

  ;   PEAK PERIOD CARPOOL3 ALONE ELEMENTS OF UTILITY ARE:
  ;   WALK TIME
      MW[11]=(MI.5.TERMINALTIME)*HBWCOVT
  ;   WAIT TIME
      ;MW[12]=(0)*HBWCOVT
  ;   IVTT
      MW[13]=(MI.5.TIME)*HBWCIVT
  ;   PARKING COST - ONLY AT DESTINATION (J), HALF IN EACH DIRECTION, SHARED BY {hbw3p}
      MW[14]=0.5 * ZI.1.LONGPARK[J] * HBWCCST/{hbw3p}
  ;   OTHER COST
      MW[15]=MI.5.DISTANCE*{HWYOPCOST} * 100 *HBWCCST/{hbw3p}
  ;   COMPOSITE UTILITY
      MW[023]=(MW[11]+ MW[13]+MW[14]+MW[15]+K1_NC_CX)/NESTMOTOR
      MW[033]=(MW[11]+ MW[13]+MW[14]+MW[15]+K1_WC_CX)/NESTMOTOR
      MW[043]=(MW[11]+ MW[13]+MW[14]+MW[15]+K1_ST_CX)/NESTMOTOR

  ;   PEAK PERIOD WALK TO LOCAL BUS ELEMENTS OF UTILITY ARE:
  ;   WALK TIME
      MW[11]=(mi.3.pkwktimelb)*HBWCOVT
  ;   WAIT TIME
      MW[12]=(mi.3.pkwttimelb)*HBWCOVT
```

```
  ;    IVTT
       MW[13]=(mi.3.pkivtimelb)*HBWCIVT
       if (mw[13]=0) mw[13]=-9999
  ;    PARKING COST
       MW[14]=(mi.3.pkpkcostlb)*HBWCCST
  ;    OTHER COST - t
       MW[15]=(mi.3.pkopcostlb*100*0.25*busfarefac)*HBWCCST
                                                    ; CS applied 25% (discounted) bus fare
                                                    ; due to employee pass program
  ;    PEDESTRIAN ENVIRONMENT
       MW[16]=HBWPTI * ZI.1.SUM[I]*0.25
  ;    COMPOSITE UTILITY
       MW[024]=(MW[11]+MW[12]+MW[13]+MW[14]+MW[15]+MW[16]+K1_NC_WB)/NESTTRANSIT
       MW[034]=(MW[11]+MW[12]+MW[13]+MW[14]+MW[15]+MW[16]+K1_WC_WB)/NESTTRANSIT
       MW[044]=(MW[11]+MW[12]+MW[13]+MW[14]+MW[15]+MW[16]+K1_ST_WB)/NESTTRANSIT

  ;    PEAK PERIOD WALK TO PREMIUM TRANSIT ELEMENTS OF UTILITY ARE:
  ;    WALK TIME
       MW[11]=(mi.3.pkwktimeex)*HBWCOVT
  ;    WAIT TIME
       MW[12]=(mi.3.pkwttimeex)*HBWCOVT
  ;    IVTT
       MW[13]=(mi.3.pkivtimeex)*HBWCIVT
       if (mw[13]=0) mw[13]=-9999
  ;    PARKING COST
       MW[14]=(mi.3.pkpkcostex)*HBWCCST
  ;    OTHER COST - FARE
       MW[15]=(mi.3.pkopcostex * 100)*HBWCCST
  ;    PEDESTRIAN ENVIRONMENT
       MW[16]=HBWPTI * ZI.1.SUM[I]*0.25
  ;    COMPOSITE UTILITY
       MW[025]=(MW[11]+MW[12]+MW[13]+MW[14]+MW[15]+MW[16]+K1_NC_WX)/NESTTRANSIT
       MW[035]=(MW[11]+MW[12]+MW[13]+MW[14]+MW[15]+MW[16]+K1_WC_WX)/NESTTRANSIT
       MW[045]=(MW[11]+MW[12]+MW[13]+MW[14]+MW[15]+MW[16]+K1_ST_WX)/NESTTRANSIT

  ;    PEAK PERIOD AUTO TO BEST AVAILABLE TRANSIT ELEMENTS OF UTILITY ARE:
  ;    WALK TIME
       MW[11]=(mi.3.pkwktimeba)*HBWCOVT
  ;    WAIT TIME
       MW[12]=(mi.3.pkwttimeba)*HBWCOVT
  ;    IVTT
       MW[13]=(mi.3.pkivtimeba)*HBWCIVT
       if (mw[13]=0) mw[13]=-9999
  ;    PARKING COST
       MW[14]=(mi.3.pkpkcostba)*HBWCCST
  ;    OTHER COST - FARE
       MW[15]=(mi.3.pkopcostba * 100)*HBWCCST
  ;    COMPOSITE UTILITY
       MW[026]=(MW[11]+MW[12]+MW[13]+MW[14]+MW[15]+K1_NC_BA)/NESTTRANSIT
       MW[036]=(MW[11]+MW[12]+MW[13]+MW[14]+MW[15]+K1_WC_BA)/NESTTRANSIT
       MW[046]=(MW[11]+MW[12]+MW[13]+MW[14]+MW[15]+K1_ST_BA)/NESTTRANSIT

; ====================================================================
; HBO, NHB, HBU AND HDORMU (OFF-PEAK) TRIP PURPOSES
; ====================================================================
  ;    OFF-PEAK PERIOD DRIVE ALONE ELEMENTS OF UTILITY ARE:
  ;    -- HBO --
  ;    WALK TIME
       MW[11]=(MI.5.TERMINALTIME)*HBOCOVT
  ;    WAIT TIME
       ;MW[12]=(0)*HBOCOVT
  ;    IVTT
       MW[13]=(MI.1.TIME)*HBOCIVT
  ;    PARKING COST - ONLY AT DESTINATION (J), HALF IN EACH DIRECTION
       MW[14]=0.5 * ZI.1.SHORTPARK[J] * HBOCCST
  ;    OTHER COST
       MW[15]=MI.1.DISTANCE * {HWYOPCOST} * 100 * HBOCCST
  ;    COMPOSITE UTILITY
       ;MW[051]=(MW[11]+MW[13]+MW[14]+MW[15]+K2_NC_DA)/NESTMOTOR
       MW[061]=(MW[11]+MW[13]+MW[14]+MW[15]+K2_WC_DA)/NESTMOTOR
       MW[071]=(MW[11]+MW[13]+MW[14]+MW[15]+K2_ST_DA)/NESTMOTOR
```

```
;    -- NHB --
;    WALK TIME
     MW[11]=(MI.5.TERMINALTIME)*NHBCOVT
;    WAIT TIME
     ;MW[12]=(0)*NHBCOVT
;    IVTT
     MW[13]=(MI.1.TIME)*NHBCIVT
;    PARKING COST - AVG ORIGIN AND DESTINATION
     MW[14]=0.5 * (ZI.1.SHORTPARK[I]+ZI.1.SHORTPARK[J]) * NHBCCST
;    OTHER COST
     MW[15]=MI.1.DISTANCE * {HWYOPCOST} * 100 * NHBCCST
;    COMPOSITE UTILITY
     MW[081]=(MW[11]+MW[13]+MW[14]+MW[15]+K3_NC_DA)/NESTMOTOR

;    -- HBU --
;    WALK TIME
     MW[11]=(MI.5.TERMINALTIME)*UNICOVT
;    WAIT TIME
     ;MW[12]=(0)*UNICOVT
;    IVTT
     MW[13]=(MI.1.TIME)*UNICIVT
;    PARKING COST - ONLY AT DESTINATION (J), HALF IN EACH DIRECTION
     MW[14]=0.5 * ZI.1.STUDENTPAR[J] * UNICCST ; university gets long term cost
;    OTHER COST
     MW[15]=MI.1.DISTANCE * {HWYOPCOST} * 100 * UNICCST
;    COMPOSITE UTILITY
     MW[101]=(MW[11]+MW[13]+MW[14]+MW[15]+K4_NC_DA)/NESTMOTOR

;    OFF-PEAK PERIOD CARPOOL2 ELEMENTS OF UTILITY ARE:
;    -- HBO --
;    WALK TIME
     MW[11]=(MI.5.TERMINALTIME)*HBOCOVT
;    WAIT TIME

     ;MW[12]=(0)*HBOCOVT
;    IVTT
     MW[13]=(MI.1.TIME)*HBOCIVT
;    PARKING COST - ONLY AT DESTINATION (J), HALF IN EACH DIRECTION

     MW[14]=0.25 * ZI.1.SHORTPARK[J] * HBOCCST
;    OTHER COST
     MW[15]=0.5 * MI.1.DISTANCE * {HWYOPCOST} * 100 * HBOCCST
;    COMPOSITE UTILITY
     MW[052]=(MW[11]+MW[13]+MW[14]+MW[15]+K2_NC_CP)/NESTMOTOR
     MW[062]=(MW[11]+MW[13]+MW[14]+MW[15]+K2_WC_CP)/NESTMOTOR
     MW[072]=(MW[11]+MW[13]+MW[14]+MW[15]+K2_ST_CP)/NESTMOTOR

;    -- NHB --
;    WALK TIME
     MW[11]=(MI.5.TERMINALTIME)*NHBCOVT
;    WAIT TIME
     ;MW[12]=(0)*NHBCOVT
;    IVTT
     MW[13]=(MI.1.TIME)*NHBCIVT
;    PARKING COST - ONLY AT DESTINATION (J), HALF IN EACH DIRECTION
     MW[14]=0.25 * (ZI.1.SHORTPARK[I]+ZI.1.SHORTPARK[J]) * NHBCCST
;    OTHER COST
     MW[15]=0.5 * MI.1.DISTANCE * {HWYOPCOST} * 100 * NHBCCST
;    COMPOSITE UTILITY
     MW[082]=(MW[11]+MW[13]+MW[14]+MW[15]+K3_NC_CP)/NESTMOTOR
;    -- HBU --
;    WALK TIME
     MW[11]=(MI.5.TERMINALTIME)*UNICOVT
;    WAIT TIME
     ;MW[12]=(0)*UNICOVT
;    IVTT
     MW[13]=(MI.1.TIME)*UNICIVT
;    PARKING COST - ONLY AT DESTINATION (J), HALF IN EACH DIRECTION
     MW[14]=0.25 * ZI.1.STUDENTPAR[J] * UNICCST ; university gets long term cost
;    OTHER COST
```

```
    MW[15]=0.5 * MI.1.DISTANCE * {HWYOPCOST} * 100 * UNICCST
;   COMPOSITE UTILITY
    MW[102]=(MW[11]+MW[13]+MW[14]+MW[15]+K4_NC_CP)/NESTMOTOR

;   OFF-PEAK PERIOD CARPOOL3 ELEMENTS OF UTILITY ARE:
;   -- HBO --
;   WALK TIME
    MW[11]=(MI.5.TERMINALTIME)*HBOCOVT
;   WAIT TIME
    ;MW[12]=(0)*HBOCOVT
;   IVTT
    MW[13]=(MI.1.TIME)*HBOCIVT
;   PARKING COST
    MW[14]=0.50 * ZI.1.SHORTPARK[J] * HBOCCST/{hbo3p}
;   OTHER COST
    MW[15]=(MI.1.DISTANCE*{HWYOPCOST} * 100)*HBOCCST/{hbo3p}
;   COMPOSITE UTILITY
    MW[053]=(MW[11]+MW[13]+MW[14]+MW[15]+K2_NC_CX)/NESTMOTOR
    MW[063]=(MW[11]+MW[13]+MW[14]+MW[15]+K2_WC_CX)/NESTMOTOR
    MW[073]=(MW[11]+MW[13]+MW[14]+MW[15]+K2_ST_CX)/NESTMOTOR

;   -- NHB --
;   WALK TIME
    MW[11]=(MI.5.TERMINALTIME)*NHBCOVT
;   WAIT TIME
    ;MW[12]=(0)*NHBCOVT
;   IVTT
    MW[13]=(MI.1.TIME)*NHBCIVT
;   PARKING COST
    MW[14]=0.50 * (ZI.1.SHORTPARK[I]+ZI.1.SHORTPARK[J]) * NHBCCST/{NHB3P}
;   OTHER COST
    MW[15]=(MI.1.DISTANCE*{HWYOPCOST} * 100)*NHBCCST/{NHB3P}
;   COMPOSITE UTILITY
    MW[083]=(MW[11]+MW[13]+MW[14]+MW[15]+K3_NC_CX)/NESTMOTOR

;   -- HBU --
;   WALK TIME
    MW[11]=(MI.5.TERMINALTIME)*UNICOVT
;   WAIT TIME
    ;MW[12]=(0)*UNICOVT
;   IVTT
    MW[13]=(MI.1.TIME)*UNICIVT
;   PARKING COST
    MW[14]=0.50 * ZI.1.STUDENTPAR[J] * UNICCST/{hbw3p} ; assume 3+ occ like work & Long park cost
;   OTHER COST
    MW[15]=(MI.1.DISTANCE*{HWYOPCOST} * 100)*UNICCST/{hbw3p} ; assume 3+ occ like work
;   COMPOSITE UTILITY
    MW[103]=(MW[11]+MW[13]+MW[14]+MW[15]+K4_NC_CX)/NESTMOTOR

;   OFF-PEAK PERIOD WALK TO LOCAL BUS ELEMENTS OF UTILITY ARE:
;   -- HBO --
;   WALK TIME
    MW[11]=(mi.4.opwktimelb)*HBOCOVT
;   WAIT TIME
    MW[12]=(mi.4.opwttimelb)*HBOCOVT
;   IVTT
    MW[13]=(mi.4.opivtimelb)*HBOCIVT
    if (mw[13]=0) mw[13]=-9999
;   PARKING COST
    MW[14]=(mi.4.oppkcostlb)*HBOCCST
;   OTHER COST - FARE
    MW[15]=(mi.4.opopcostlb * 100 * busfarefac)*HBOCCST
;   PEDESTRIAN ENVIRONMENT
    MW[16]=HBOPTI * ZI.1.SUM[I]*0.25
;   COMPOSITE UTILITY
    MW[054]=(MW[11]+MW[12]+MW[13]+MW[14]+MW[15]+MW[16]+K2_NC_WB)/NESTTRANSIT
    MW[064]=(MW[11]+MW[12]+MW[13]+MW[14]+MW[15]+MW[16]+K2_WC_WB)/NESTTRANSIT
    MW[074]=(MW[11]+MW[12]+MW[13]+MW[14]+MW[15]+MW[16]+K2_ST_WB)/NESTTRANSIT

;   -- NHB --
;   WALK TIME
```

```
        MW[11]=(mi.4.opwktimelb)*NHBCOVT
    ;   WAIT TIME
        MW[12]=(mi.4.opwttimelb)*NHBCOVT
    ;   IVTT
        MW[13]=(mi.4.opivtimelb)*NHBCIVT
        if (mw[13]=0) mw[13]=-9999
    ;   PARKING COST
        MW[14]=(mi.4.oppkcostlb)*NHBCCST
    ;   OTHER COST - FARE
        MW[15]=(mi.4.opopcostlb * 100 * busfarefac)*NHBCCST
    ;   PEDESTRIAN ENVIRONMENT
        MW[16]=NHBPTI * ZI.1.SUM[I]*0.25
    ;   COMPOSITE UTILITY
        MW[084]=(MW[11]+MW[12]+MW[13]+MW[14]+MW[15]+MW[16]+K3_NC_WB)/NESTTRANSIT

    ;   -- HBU --
    ;   WALK TIME
        MW[11]=(mi.4.opwktimelb)*UNICOVT
    ;   WAIT TIME
        MW[12]=(mi.4.opwttimelb)*UNICOVT
    ;   IVTT
        MW[13]=(mi.4.opivtimelb)*UNICIVT
        if (mw[13]=0) mw[13]=-9999
    ;   PARKING COST
        MW[14]=(mi.4.oppkcostlb)*UNICCST
    ;   OTHER COST - FARE
        MW[15]=(mi.4.opopcostlb * 100 * 0.10*busfarefac)*UNICCST
                                                    ; CS applied 10% (discounted) bus fare
                                                    ; due to transit fare payed in tuition
;       MW[15]=0 ; UF fare free - previous model
    ;   PEDESTRIAN ENVIRONMENT
        MW[16]=UNIPTI * ZI.1.SUM[I]*0.25
    ;   COMPOSITE UTILITY
        MW[104]=(MW[11]+MW[12]+MW[13]+MW[14]+MW[15]+MW[16]+K4_NC_WB)/NESTTRANSIT

    ;   -- HDORMU --
    ;   WALK TIME
        MW[11]=(mi.4.opwktimelb)*UNICOVT
    ;   WAIT TIME
        MW[12]=(mi.4.opwttimelb)*UNICOVT
    ;   IVTT
        MW[13]=(mi.4.opivtimelb)*UNICIVT
        if (mw[13]=0) mw[13]=-9999
    ;   PARKING COST
        MW[14]=(mi.4.oppkcostlb)*UNICCST
    ;   OTHER COST - FARE
        MW[15]=(mi.4.opopcostlb * 100 * 0.10*busfarefac)*UNICCST
                                                    ; CS applied 10% (discounted) bus fare
                                                    ; due to transit fare payed in tuition
        ;MW[15]=0 ; UF fare free - previous model
    ;   PEDESTRIAN ENVIRONMENT
        MW[16]=UNIPTI * ZI.1.SUM[I]*0.25
    ;   COMPOSITE UTILITY
        MW[110]=(MW[11]+MW[12]+MW[13]+MW[14]+MW[15]+MW[16]+K5_NC_WB);/{NESTCTRANSIT}

    ;   OFF-PEAK PERIOD WALK TO PREMIUM TRANSIT ELEMENTS OF UTILITY ARE:

    ;   -- HBO --
    ;   WALK TIME
        MW[11]=(mi.4.opwktimeex)*HBOCOVT
    ;   WAIT TIME
        MW[12]=(mi.4.opwttimeex)*HBOCOVT
    ;   IVTT
        MW[13]=(mi.4.opivtimeex)*HBOCIVT
        if (mw[13]=0) mw[13]=-9999
    ;   PARKING COST
        MW[14]=(mi.4.oppkcostex)*HBOCCST
    ;   OTHER COST
        MW[15]=(mi.4.opopcostex * 100)*HBOCCST
    ;   PEDESTRIAN ENVIRONMENT
        MW[16]=HBOPTI * ZI.1.SUM[I]*0.25
```

```
;   COMPOSITE UTILITY
    MW[055]=(MW[11]+MW[12]+MW[13]+MW[14]+MW[15]+MW[16]+K2_NC_WX)/NESTTRANSIT
    MW[065]=(MW[11]+MW[12]+MW[13]+MW[14]+MW[15]+MW[16]+K2_WC_WX)/NESTTRANSIT
    MW[075]=(MW[11]+MW[12]+MW[13]+MW[14]+MW[15]+MW[16]+K2_ST_WX)/NESTTRANSIT

;   -- NHB --
;   WALK TIME
    MW[11]=(mi.4.opwktimeex)*NHBCOVT
;   WAIT TIME
    MW[12]=(mi.4.opwttimeex)*NHBCOVT
;   IVTT
    MW[13]=(mi.4.opivtimeex)*NHBCIVT
    if (mw[13]=0) mw[13]=-9999
;   PARKING COST
    MW[14]=(mi.4.oppkcostex)*NHBCCST
;   OTHER COST
    MW[15]=(mi.4.opopcostex * 100)*NHBCCST
;   PEDESTRIAN ENVIRONMENT
    MW[16]=NHBPTI * ZI.1.SUM[I]*0.25
;   COMPOSITE UTILITY
    MW[085]=(MW[11]+MW[12]+MW[13]+MW[14]+MW[15]+MW[16]+K3_NC_WX)/NESTTRANSIT

;   -- HBU --
;   WALK TIME
    MW[11]=(mi.4.opwktimeex)*UNICOVT
;   WAIT TIME
    MW[12]=(mi.4.opwttimeex)*UNICOVT
;   IVTT
    MW[13]=(mi.4.opivtimeex)*UNICIVT
    if (mw[13]=0) mw[13]=-9999
;   PARKING COST
    MW[14]=(mi.4.oppkcostex)*UNICCST
;   OTHER COST
    ;MW[15]=(mi.4.opopcostex)*UNICCST
    MW[15]=0 ; UF fare free
;   PEDESTRIAN ENVIRONMENT
    MW[16]=UNIPTI * ZI.1.SUM[I]*0.25
;   COMPOSITE UTILITY
    MW[105]=(MW[11]+MW[12]+MW[13]+MW[14]+MW[15]+MW[16]+K4_NC_WX)/NESTTRANSIT

;   OFF-PEAK PERIOD AUTO TO BEST AVAILABLE TRANSIT ELEMENTS OF UTILITY ARE:
;   -- HBO --
;   WALK TIME
    MW[11]=(mi.4.opwktimeba)*HBOCOVT
;   WAIT TIME
    MW[12]=(mi.4.opwttimeba)*HBOCOVT
;   IVTT
    MW[13]=(mi.4.opivtimeba)*HBOCIVT
    if (mw[13]=0) mw[13]=-9999
;   PARKING COST
    MW[14]=(mi.4.oppkcostba)*HBOCCST
;   OTHER COST
    MW[15]=(mi.4.opopcostba * 100)*HBOCCST
;   COMPOSITE UTILITY
    MW[056]=(MW[11]+MW[12]+MW[13]+MW[14]+MW[15]+K2_NC_BA)/NESTTRANSIT
    MW[066]=(MW[11]+MW[12]+MW[13]+MW[14]+MW[15]+K2_WC_BA)/NESTTRANSIT
    MW[076]=(MW[11]+MW[12]+MW[13]+MW[14]+MW[15]+K2_ST_BA)/NESTTRANSIT

;   -- NHB --
;   WALK TIME
    MW[11]=(mi.4.opwktimeba)*NHBCOVT
;   WAIT TIME
    MW[12]=(mi.4.opwttimeba)*NHBCOVT
;   IVTT
    MW[13]=(mi.4.opivtimeba)*NHBCIVT
    if (mw[13]=0) mw[13]=-9999
;   PARKING COST
    MW[14]=(mi.4.oppkcostba)*NHBCCST
;   OTHER COST
    MW[15]=(mi.4.opopcostba * 100)*NHBCCST
;   COMPOSITE UTILITY
```

```
      MW[086]=(MW[11]+MW[12]+MW[13]+MW[14]+MW[15]+K3_NC_BA)/NESTTRANSIT

  ;   -- HBU --
  ;   WALK TIME
      MW[11]=(mi.4.opwktimeba)*UNICOVT
  ;   WAIT TIME
      MW[12]=(mi.4.opwttimeba)*UNICOVT
  ;   IVTT
      MW[13]=(mi.4.opivtimeba)*UNICIVT
      if (mw[13]=0) mw[13]=-9999
  ;   PARKING COST
      MW[14]=(mi.4.oppkcostba)*UNICCST
  ;   OTHER COST
      ;MW[15]=(mi.4.opopcostba)*UNICCST
      MW[15]=0 ; UF fare free
  ;   COMPOSITE UTILITY
      MW[106]=(MW[11]+MW[12]+MW[13]+MW[14]+MW[15]+K4_NC_BA)/NESTTRANSIT

; ---------- END MOTORIZED UTILITIES --------------------

; WALK ONLY (NON-MOTORIZED) ELEMENTS OF UTILITY ARE:
; WALK AND BIKE TIMES
      mw[8]=60*MI.1.WALKDISTANCE/{WALKSPEED} ;all walk
      mw[9]=MI.1.BIKETIME ;all bike
;HBW
  ;   WALK TIME
      MW[11]=mw[8]*HBWCWT
  ;   PEDESTRIAN ENVIRONMENT
      MW[12]= 0.25*(ZI.1.SUM[I]*HBWPWI + ZI.1.SUM[J]*HBWPWJ)
      ; 0.25 because we are using sum, not composite
  ;   UTILITIES
      MW[027]=(MW[11]+MW[12]+K1_NC_WK)/{NESTCNONMOTOR}
      MW[037]=(MW[11]+MW[12]+K1_WC_WK)/{NESTCNONMOTOR}
      MW[047]=(MW[11]+MW[12]+K1_ST_WK)/{NESTCNONMOTOR}
;HBO
  ;   WALK TIME
      MW[11]=mw[8]*HBOCWT
  ;   PEDESTRIAN ENVIRONMENT
      MW[12]= 0.25*(ZI.1.SUM[I]*HBOPWI + ZI.1.SUM[J]*HBOPWJ)
  ;   UTILITIES
      MW[057]=(MW[11]+MW[12]+K2_NC_WK)/{NESTCNONMOTOR}
      MW[067]=(MW[11]+MW[12]+K2_WC_WK)/{NESTCNONMOTOR}
      MW[077]=(MW[11]+MW[12]+K2_ST_WK)/{NESTCNONMOTOR}
;NHB
  ;   WALK TIME
      MW[11]=mw[8]*NHBCWT
  ;   PEDESTRIAN ENVIRONMENT
      MW[12]= 0.25*(ZI.1.SUM[I]*NHBPWI + ZI.1.SUM[J]*NHBPWJ)
  ;   UTILITIES
      MW[087]=(MW[11]+MW[12]+K3_NC_WK)/{NESTCNONMOTOR}
;UNIVERSITY
  ;   WALK TIME
      MW[11]=mw[8]*UNICWT
  ;   PEDESTRIAN ENVIRONMENT
      MW[12]= 0.25*(ZI.1.SUM[I]*UNIPWI + ZI.1.SUM[J]*UNIPWJ)
  ;   UTILITIES
      MW[089]=(MW[11]+MW[12]+K4_NC_WK)/{NESTCNONMOTOR}
      MW[090]=(MW[11]+MW[12]+K5_NC_WK);/{NESTCNONMOTOR}

; BIKE ONLY (NON-MOTORIZED) ELEMENTS OF UTILITY ARE:

;HBW
  ;   BIKE TIME
      MW[11]=mw[9]*HBWCBT
  ;   PEDESTRIAN ENVIRONMENT
      MW[12]=0.25*(ZI.1.SUM[I]*HBWPBI + ZI.1.SUM[J]*HBWPBJ)
  ;   UTILITIES
      MW[028]=(MW[11]+MW[12]+K1_NC_BK)/{NESTCNONMOTOR}
      MW[038]=(MW[11]+MW[12]+K1_WC_BK)/{NESTCNONMOTOR}
      MW[048]=(MW[11]+MW[12]+K1_ST_BK)/{NESTCNONMOTOR}
;HBO
```

```
  ;   BIKE TIME
      MW[11]=mw[9]*HBOCBT
  ;   PEDESTRIAN ENVIRONMENT
      MW[12]=0.25*(ZI.1.SUM[I]*HBOPBI + ZI.1.SUM[J]*HBOPBJ)
  ;   UTILITIES
      MW[058]=(MW[11]+MW[12]+K2_NC_BK)/{NESTCNONMOTOR}
      MW[068]=(MW[11]+MW[12]+K2_WC_BK)/{NESTCNONMOTOR}
      MW[078]=(MW[11]+MW[12]+K2_ST_BK)/{NESTCNONMOTOR}
;NHB
  ;   BIKE TIME
      MW[11]=mw[9]*NHBCBT
  ;   PEDESTRIAN ENVIRONMENT
      MW[12]=0.25*(ZI.1.SUM[I]*NHBPBI + ZI.1.SUM[J]*NHBPBJ)
  ;   UTILITIES
      MW[088]=(MW[11]+MW[12]+K3_NC_BK)/{NESTCNONMOTOR}
;UNIVERSITY
  ;   BIKE TIME
      MW[11]=mw[9]*UNICBT
  ;   PEDESTRIAN ENVIRONMENT
      MW[12]=0.25*(ZI.1.SUM[I]*UNIPBI + ZI.1.SUM[J]*UNIPBJ)
  ;   UTILITIES
      MW[091]=(MW[11]+MW[12]+K4_NC_BK)/{NESTCNONMOTOR}
      MW[092]=(MW[11]+MW[12]+K5_NC_BK);/{NESTCNONMOTOR}
    endjloop

; MARKET SEGMENTATION: car, no car student
     MW[301]=MW[1]*ZI.3.NOCARPCT ; 0 car
     MW[302]=MW[1]*ZI.3.WCARPCT  ; with car
     MW[303]=MW[1]*ZI.3.STUPCT   ; students
     MW[304]=MW[2]*ZI.3.NOCARPCT ; 0 car
     MW[305]=MW[2]*ZI.3.WCARPCT  ; with car
     MW[306]=MW[2]*ZI.3.STUPCT   ; students

; HBW (USE 0 CAR)
; DA, BA not in market
   CHOICE ALTERNATIVES=CP,CX,WB,WX,WK,BK,
      DEMAND=MW[301],
      UTILITIES=MW[022],MW[023],MW[024],MW[025],MW[027],MW[028],
      ODEMAND=402,403,404,405,407,408,
      STARTMW=500,
      SPLIT=TOTAL,    {NESTCNONMOTOR} NONMOTOR, {NESTCMOTOR} MOTOR,
      SPLIT=MOTOR,    {NESTCAUTO} AUTO,     {NESTCTRANSIT} TRANSIT,
      SPLIT=NONMOTOR, 1.0 WK, 1.0 BK,
      SPLIT=AUTO,     1.0 CP, 1.0 CX,
      SPLIT=TRANSIT,  1.0 WB, 1.0 WX
  MW[401]=0; no drive alone
  MW[406]=0; no auto access

; HBW (USE 1+ CAR)
   CHOICE ALTERNATIVES=DA,CP,CX,WB,WX,BA,WK,BK,
      DEMAND=MW[302],
      UTILITIES=MW[031],MW[032],MW[033],MW[034],MW[035],MW[036],MW[037],MW[038],
      ODEMAND=411,412,413,414,415,416,417,418,
      STARTMW=500,
      SPLIT=TOTAL,    {NESTCNONMOTOR} NONMOTOR, {NESTCMOTOR} MOTOR,
      SPLIT=MOTOR,    {NESTCAUTO} AUTO,     {NESTCTRANSIT} TRANSIT,
      SPLIT=NONMOTOR, 1.0 WK, 1.0 BK,
      SPLIT=AUTO,     1.0 DA, 1.0 CP, 1.0 CX,
      SPLIT=TRANSIT,  1.0 WB, 1.0 WX, 1.0 BA
; HBW (USE STUDENT)
   CHOICE ALTERNATIVES=DA,CP,CX,WB,WX,BA,WK,BK,
      DEMAND=MW[303],
      UTILITIES=MW[041],MW[042],MW[043],MW[044],MW[045],MW[046],MW[047],MW[048],
      ODEMAND=421,422,423,424,425,426,427,428,
      STARTMW=500,
      SPLIT=TOTAL,    {NESTCNONMOTOR} NONMOTOR, {NESTCMOTOR} MOTOR,
      SPLIT=MOTOR,    {NESTCAUTO} AUTO,     {NESTCTRANSIT} TRANSIT,
      SPLIT=NONMOTOR, 1.0 WK, 1.0 BK,
      SPLIT=AUTO,     1.0 DA, 1.0 CP, 1.0 CX,
      SPLIT=TRANSIT,  1.0 WB, 1.0 WX, 1.0 BA
MW[151]=MW[401]+MW[411]+MW[421]
```

```
MW[152]=MW[402]+MW[412]+MW[422]
MW[153]=MW[403]+MW[413]+MW[423]
MW[154]=MW[404]+MW[414]+MW[424]
MW[155]=MW[405]+MW[415]+MW[425]
MW[156]=MW[406]+MW[416]+MW[426]
MW[157]=MW[407]+MW[417]+MW[427]
MW[158]=MW[408]+MW[418]+MW[428]

; HBO (USE 0 CAR)
; DA, BA not in market
   CHOICE ALTERNATIVES=CP,CX,WB,WX,WK,BK,
      DEMAND=MW[304],
      UTILITIES=MW[052],MW[053],MW[054],MW[055],MW[057],MW[058],
      ODEMAND=432,433,434,435,437,438,
      STARTMW=500,
      SPLIT=TOTAL,    {NESTCNONMOTOR} NONMOTOR, {NESTCMOTOR} MOTOR,
      SPLIT=MOTOR,    {NESTCAUTO} AUTO,     {NESTCTRANSIT} TRANSIT,
      SPLIT=NONMOTOR, 1.0 WK, 1.0 BK,
      SPLIT=AUTO,     1.0 CP, 1.0 CX,
      SPLIT=TRANSIT,  1.0 WB, 1.0 WX

  MW[431]=0; no drive alone
  MW[436]=0; no auto access

; HBO (USE 1+ CAR)
   CHOICE ALTERNATIVES=DA,CP,CX,WB,WX,BA,WK,BK,
      DEMAND=MW[305],
      UTILITIES=MW[061],MW[062],MW[063],MW[064],MW[065],MW[066],MW[067],MW[068],
      ODEMAND=441,442,443,444,445,446,447,448,
      STARTMW=500,
      SPLIT=TOTAL,    {NESTCNONMOTOR} NONMOTOR, {NESTCMOTOR} MOTOR,
      SPLIT=MOTOR,    {NESTCAUTO} AUTO,     {NESTCTRANSIT} TRANSIT,
      SPLIT=NONMOTOR, 1.0 WK, 1.0 BK,
      SPLIT=AUTO,     1.0 DA, 1.0 CP, 1.0 CX,
      SPLIT=TRANSIT,  1.0 WB, 1.0 WX, 1.0 BA
; HBO (USE STUDENT)
   CHOICE ALTERNATIVES=DA,CP,CX,WB,WX,BA,WK,BK,
      DEMAND=MW[306],
      UTILITIES=MW[071],MW[072],MW[073],MW[074],MW[075],MW[076],MW[077],MW[078],
      ODEMAND=451,452,453,454,455,456,457,458,
      STARTMW=500,
      SPLIT=TOTAL,    {NESTCNONMOTOR} NONMOTOR, {NESTCMOTOR} MOTOR,
      SPLIT=MOTOR,    {NESTCAUTO} AUTO,     {NESTCTRANSIT} TRANSIT,
      SPLIT=NONMOTOR, 1.0 WK, 1.0 BK,
      SPLIT=AUTO,     1.0 DA, 1.0 CP, 1.0 CX,
      SPLIT=TRANSIT,  1.0 WB, 1.0 WX, 1.0 BA
MW[161]=MW[431]+MW[441]+MW[451]
MW[162]=MW[432]+MW[442]+MW[452]
MW[163]=MW[433]+MW[443]+MW[453]
MW[164]=MW[434]+MW[444]+MW[454]
MW[165]=MW[435]+MW[445]+MW[455]
MW[166]=MW[436]+MW[446]+MW[456]
MW[167]=MW[437]+MW[447]+MW[457]
MW[168]=MW[438]+MW[448]+MW[458]

; NHB (USE 0 CAR CONSTANTS, NO MARKET SEGMENTATION IS NEEDED FOR THIS TRIP PURPOSE)
   CHOICE ALTERNATIVES=DA,CP,CX,WB,WX,BA,WK,BK,
      DEMAND=MW[003],
      UTILITIES=MW[081],MW[082],MW[083],MW[084],MW[085],MW[086],MW[087],MW[088],
      ODEMAND=171,172,173,174,175,176,177,178,
      STARTMW=500,
      SPLIT=TOTAL,    {NESTCNONMOTOR} NONMOTOR, {NESTCMOTOR} MOTOR,
      SPLIT=MOTOR,    {NESTCAUTO} AUTO,     {NESTCTRANSIT} TRANSIT,
      SPLIT=NONMOTOR, 1.0 WK, 1.0 BK,
      SPLIT=AUTO,     1.0 DA, 1.0 CP, 1.0 CX,
      SPLIT=TRANSIT,  1.0 WB, 1.0 WX, 1.0 BA

; HBU (FULL CHOICE SET)
   CHOICE ALTERNATIVES=DA,CP,CX,WB,WX,BA,WK,BK,
      DEMAND=MW[004],
      UTILITIES=MW[101],MW[102],MW[103],MW[104],MW[105],MW[106],MW[089],MW[091],
```

```
      ODEMAND=181,182,183,184,185,186,187,188,
      STARTMW=500,
      SPLIT=TOTAL,    {NESTCNONMOTOR} NONMOTOR, {NESTCMOTOR} MOTOR,
      SPLIT=MOTOR,    {NESTCAUTO} AUTO,     {NESTCTRANSIT} TRANSIT,
      SPLIT=NONMOTOR, 1.0 WK, 1.0 BK,
      SPLIT=AUTO,     1.0 DA, 1.0 CP, 1.0 CX,
      SPLIT=TRANSIT,  1.0 WB, 1.0 WX, 1.0 BA
; CAMPUS HOUSING -HDORMU- (PARTIAL CHOICE SET)
   CHOICE ALTERNATIVES=WB,WK,BK,
      DEMAND=MW[5],
      UTILITIES=MW[110],MW[090],MW[092],
      ODEMAND=191,192,193,
      STARTMW=600,
      SPLIT=TOTAL,    1.0 WB, 1.0 WK, 1.0 BK

JLOOP
;
HBWDA=HBWDA+MW[151]
HBWCP=HBWCP+MW[152]
HBWCX=HBWCX+MW[153]
HBWWB=HBWWB+MW[154]
HBWWX=HBWWX+MW[155]
HBWBA=HBWBA+MW[156]
HBWWK=HBWWK+MW[157]
HBWBK=HBWBK+MW[158]
HBODA=HBODA+MW[161]
HBOCP=HBOCP+MW[162]
HBOCX=HBOCX+MW[163]
HBOWB=HBOWB+MW[164]
HBOWX=HBOWX+MW[165]
HBOBA=HBOBA+MW[166]
HBOWK=HBOWK+MW[167]
HBOBK=HBOBK+MW[168]
NHBDA=NHBDA+MW[171]
NHBCP=NHBCP+MW[172]
NHBCX=NHBCX+MW[173]
NHBWB=NHBWB+MW[174]
NHBWX=NHBWX+MW[175]
NHBBA=NHBBA+MW[176]
NHBWK=NHBWK+MW[177]
NHBBK=NHBBK+MW[178]
HBUDA=HBUDA+MW[181]
HBUCP=HBUCP+MW[182]
HBUCX=HBUCX+MW[183]
HBUWB=HBUWB+MW[184]
HBUWX=HBUWX+MW[185]
HBUBA=HBUBA+MW[186]
HBUWK=HBUWK+MW[187]
HBUBK=HBUBK+MW[188]
UNIWB=UNIWB+MW[191]
UNIWK=UNIWK+MW[192]
UNIBK=UNIBK+MW[193]
MW401=MW401+MW[401]
MW411=MW411+MW[411]
MW421=MW421+MW[421]
MW431=MW431+MW[431]
MW441=MW441+MW[441]
MW451=MW451+MW[451]
MW171=MW171+MW[171]
MW181=MW181+MW[181]
MW402=MW402+MW[402]
MW412=MW412+MW[412]
MW422=MW422+MW[422]
MW432=MW432+MW[432]
MW442=MW442+MW[442]
MW452=MW452+MW[452]
MW172=MW172+MW[172]
MW182=MW182+MW[182]
MW403=MW403+MW[403]
MW413=MW413+MW[413]
MW423=MW423+MW[423]
```

```
MW433=MW433+MW[433]
MW443=MW443+MW[443]
MW453=MW453+MW[453]
MW173=MW173+MW[173]
MW183=MW183+MW[183]
MW404=MW404+MW[404]
MW414=MW414+MW[414]
MW424=MW424+MW[424]
MW434=MW434+MW[434]
MW444=MW444+MW[444]
MW454=MW454+MW[454]
MW174=MW174+MW[174]
MW184=MW184+MW[184]
MW191=MW191+MW[191]
MW405=MW405+MW[405]
MW415=MW415+MW[415]
MW425=MW425+MW[425]
MW435=MW435+MW[435]
MW445=MW445+MW[445]
MW455=MW455+MW[455]
MW175=MW175+MW[175]
MW185=MW185+MW[185]
MW406=MW406+MW[406]
MW416=MW416+MW[416]
MW426=MW426+MW[426]
MW436=MW436+MW[436]
MW446=MW446+MW[446]
MW456=MW456+MW[456]
MW176=MW176+MW[176]
MW186=MW186+MW[186]
MW407=MW407+MW[407]
MW417=MW417+MW[417]
MW427=MW427+MW[427]
MW437=MW437+MW[437]
MW447=MW447+MW[447]
MW457=MW457+MW[457]
MW177=MW177+MW[177]
MW187=MW187+MW[187]
MW192=MW192+MW[192]
MW408=MW408+MW[408]
MW418=MW418+MW[418]
MW428=MW428+MW[428]
MW438=MW438+MW[438]
MW448=MW448+MW[448]
MW458=MW458+MW[458]
MW178=MW178+MW[178]
MW188=MW188+MW[188]
MW193=MW193+MW[193]
ENDJLOOP
if (i=_zones)

SUMHBW=HBWDA+HBWCP+HBWCX+HBWWB+HBWWX+HBWBA+HBWWK+HBWBK
SUMHBO=HBODA+HBOCP+HBOCX+HBOWB+HBOWX+HBOBA+HBOWK+HBOBK
SUMNHB=NHBDA+NHBCP+NHBCX+NHBWB+NHBWX+NHBBA+NHBWK+NHBBK
SUMHBU=HBUDA+HBUCP+HBUCX+HBUWB+HBUWX+HBUBA+HBUWK+HBUBK
SUMUNI=UNIWB+UNIWK+UNIBK
;Total trips by purpose
mx1=MW401+MW402+MW403+MW404+MW405+MW406+MW407+MW408
mx2=MW411+MW412+MW413+MW414+MW415+MW416+MW417+MW418
mx3=MW421+MW422+MW423+MW424+MW425+MW426+MW427+MW428
mx4=MW431+MW432+MW433+MW434+MW435+MW436+MW437+MW438
mx5=MW441+MW442+MW443+MW444+MW445+MW446+MW447+MW448
mx6=MW451+MW452+MW453+MW454+MW455+MW456+MW457+MW458
mx7=MW171+MW172+MW173+MW174+MW175+MW176+MW177+MW178
mx8=MW181+MW182+MW183+MW184+MW185+MW186+MW187+MW188
mx9=MW191+MW192+MW193
;mode shares
MW401=MW401/mx1
MW402=MW402/mx1
MW403=MW403/mx1
MW404=MW404/mx1
```

```
MW405=MW405/mx1
MW406=MW406/mx1
MW407=MW407/mx1
MW408=MW408/mx1
MW411=MW411/mx2
MW412=MW412/mx2
MW413=MW413/mx2
MW414=MW414/mx2
MW415=MW415/mx2
MW416=MW416/mx2
MW417=MW417/mx2
MW418=MW418/mx2
MW421=MW421/mx3
MW422=MW422/mx3
MW423=MW423/mx3
MW424=MW424/mx3
MW425=MW425/mx3
MW426=MW426/mx3
MW427=MW427/mx3
MW428=MW428/mx3
MW431=MW431/mx4
MW432=MW432/mx4
MW433=MW433/mx4
MW434=MW434/mx4
MW435=MW435/mx4
MW436=MW436/mx4
MW437=MW437/mx4
MW438=MW438/mx4
MW441=MW441/mx5
MW442=MW442/mx5
MW443=MW443/mx5
MW444=MW444/mx5
MW445=MW445/mx5
MW446=MW446/mx5
MW447=MW447/mx5
MW448=MW448/mx5
MW451=MW451/mx6
MW452=MW452/mx6
MW453=MW453/mx6
MW454=MW454/mx6
MW455=MW455/mx6
MW456=MW456/mx6
MW457=MW457/mx6
MW458=MW458/mx6
MW171=MW171/mx7
MW172=MW172/mx7
MW173=MW173/mx7
MW174=MW174/mx7
MW175=MW175/mx7
MW176=MW176/mx7
MW177=MW177/mx7
MW178=MW178/mx7
MW181=MW181/mx8
MW182=MW182/mx8
MW183=MW183/mx8
MW184=MW184/mx8
MW185=MW185/mx8
MW186=MW186/mx8
MW187=MW187/mx8
MW188=MW188/mx8
MW191=MW191/mx9
MW192=MW192/mx9
MW193=MW193/mx9

PRINT LIST="\n +++++ MODE CHOICE SUMMARY +++++\n" PRINTO=1
PRINT LIST="{DESC}" PRINTO=1
PRINT LIST="{SCENARIO_SHORTNAME}\n" PRINTO=1

PRINT FORM=8.0C, LIST='HOME-BASED WORK MODE CHOICE RESULTS',
'\nHBW TOTAL                    =',SUMHBW,'  ',1.0(5.4),
'\nDRIVE ALONE                  =',HBWDA,'  ',HBWDA/SUMHBW(5.4),
```

```
'\nCARPOOL 2                      =',HBWCP,'  ',HBWCP/SUMHBW(5.4),
'\nCARPOOL 3+                     =',HBWCX,'  ',HBWCX/SUMHBW(5.4),
'\nWALK TO LOCAL TRANSIT          =',HBWWB,'  ',HBWWB/SUMHBW(5.4),
'\nWALK TO PREMIUM TRANSIT        =',HBWWX,'  ',HBWWX/SUMHBW(5.4),
'\nDRIVE TO BEST AVAILABLE TRANSIT=',HBWBA,'  ',HBWBA/SUMHBW(5.4),
'\nNON-MOTORIZED WALK             =',HBWWK,'  ',HBWWK/SUMHBW(5.4),
'\nNON-MOTORIZED BICYCLE          =',HBWBK,'  ',HBWBK/SUMHBW(5.4),
'\n   Average Auto Occupancy     =',(HBWDA+HBWCP+HBWCX)/(HBWDA+HBWCP/2+HBWCX/{HBW3P})(4.3),PRINTO=1

PRINT FORM=8.OC, LIST='\n ','\nHOME-BASED OTHER MODE CHOICE RESULTS',
'\nHBO TOTAL                      =',SUMHBO,'  ',1.0(5.4),
'\nDRIVE ALONE                    =',HBODA,'  ',HBODA/SUMHBO(5.4),
'\nCARPOOL 2                      =',HBOCP,'  ',HBOCP/SUMHBO(5.4),
'\nCARPOOL 3+                     =',HBOCX,'  ',HBOCX/SUMHBO(5.4),
'\nWALK TO LOCAL TRANSIT          =',HBOWB,'  ',HBOWB/SUMHBO(5.4),
'\nWALK TO PREMIUM TRANSIT        =',HBOWX,'  ',HBOWX/SUMHBO(5.4),
'\nDRIVE TO BEST AVAILABLE TRANSIT=',HBOBA,'  ',HBOBA/SUMHBO(5.4),
'\nNON-MOTORIZED WALK             =',HBOWK,'  ',HBOWK/SUMHBO(5.4),
'\nNON-MOTORIZED BICYCLE          =',HBOBK,'  ',HBOBK/SUMHBO(5.4),
'\n   Average Auto Occupancy     =',(HBODA+HBOCP+HBOCX)/(HBODA+HBOCP/2+HBOCX/{HBO3P})(4.3),PRINTO=1

PRINT FORM=8.OC, LIST='\n ','\nNON-HOME BASED MODE CHOICE RESULTS',
'\nNHB TOTAL                      =',SUMNHB,'  ',1.0(5.4),
'\nDRIVE ALONE                    =',NHBDA,'  ',NHBDA/SUMNHB(5.4),
'\nCARPOOL 2                      =',NHBCP,'  ',NHBCP/SUMNHB(5.4),
'\nCARPOOL 3+                     =',NHBCX,'  ',NHBCX/SUMNHB(5.4),
'\nWALK TO LOCAL TRANSIT          =',NHBWB,'  ',NHBWB/SUMNHB(5.4),
'\nWALK TO PREMIUM TRANIST        =',NHBWX,'  ',NHBWX/SUMNHB(5.4),
'\nDRIVE TO BEST AVAILABLE TRANSIT=',NHBBA,'  ',NHBBA/SUMNHB(5.4),
'\nNON-MOTORIZED WALK             =',NHBWK,'  ',NHBWK/SUMNHB(5.4),
'\nNON-MOTORIZED BICYCLE          =',NHBBK,'  ',NHBBK/SUMNHB(5.4),
'\n   Average Auto Occupancy     =',(NHBDA+NHBCP+NHBCX)/(NHBDA+NHBCP/2+NHBCX/{NHB3P})(4.3),PRINTO=1

PRINT FORM=8.OC, LIST='\n ','\nHOME BASED UNIVERSITY MODE CHOICE RESULTS',
'\nHBU TOTAL                      =',SUMHBU,'  ',1.0(5.4),
'\nDRIVE ALONE                    =',HBUDA,'  ',HBUDA/SUMHBU(5.4),
'\nCARPOOL 2                      =',HBUCP,'  ',HBUCP/SUMHBU(5.4),
'\nCARPOOL 3+                     =',HBUCX,'  ',HBUCX/SUMHBU(5.4),
'\nWALK TO LOCAL TRANSIT          =',HBUWB,'  ',HBUWB/SUMHBU(5.4),
'\nWALK TO PREMIUM TRANSIT        =',HBUWX,'  ',HBUWX/SUMHBU(5.4),
'\nDRIVE TO BEST AVAILABLE TRANSIT=',HBUBA,'  ',HBUBA/SUMHBU(5.4),
'\nNON-MOTORIZED WALK             =',HBUWK,'  ',HBUWK/SUMHBU(5.4),
'\nNON-MOTORIZED BICYCLE          =',HBUBK,'  ',HBUBK/SUMHBU(5.4),
'\n   Average Auto Occupancy     =',(HBUDA+HBUCP+HBUCX)/(HBUDA+HBUCP/2+HBUCX/{HBW3P})(4.3),PRINTO=1

PRINT FORM=8.OC, LIST='\n ','\nCAMPUS UNIVERSITY MODE CHOICE RESULTS',
'\nCAMPUS HOUSING TOTAL           =',SUMUNI,'  ',1.0(5.4),
'\nWALK TO LOCAL TRANSIT          =',UNIWB,'  ',UNIWB/SUMUNI(5.4),
'\nNON-MOTORIZED WALK             =',UNIWK,'  ',UNIWK/SUMUNI(5.4),
'\nNON-MOTORIZED BICYCLE          =',UNIBK,'  ',UNIBK/SUMUNI(5.4),PRINTO=1

; MODE SUMMARY TABLE AS CSV
PRINT                                       CSV=T,                                      LIST=
'PURPOSE','TOTAL','DA','SR2','SR3+','WALKBUS','WALKPREM','DRIVETRN','WALK','BIKE', PRINTO=4
PRINT CSV=T, LIST= 'HBW',SUMHBW,HBWDA,HBWCP,HBWCX,HBWWB,HBWWX,HBWBA,HBWWK,HBWBK,PRINTO=4
PRINT CSV=T, LIST= 'HBO',SUMHBO,HBODA,HBOCP,HBOCX,HBOWB,HBOWX,HBOBA,HBOWK,HBOBK,PRINTO=4
PRINT CSV=T, LIST= 'NHB',SUMNHB,NHBDA,NHBCP,NHBCX,NHBWB,NHBWX,NHBBA,NHBWK,NHBBK,PRINTO=4
PRINT CSV=T, LIST= 'HBU',SUMHBU,HBUDA,HBUCP,HBUCX,HBUWB,HBUWX,HBUBA,HBUWK,HBUBK,PRINTO=4
PRINT CSV=T, LIST= 'DORM',SUMUNI,0,0,0,UNIWB,0,0,UNIWK,UNIBK,PRINTO=4

; Targets
PRINT CSV=T, LIST= 'TARGETS', PRINTO=2
PRINT                                       CSV=T,                                      LIST=
1,t1_NC_DA(7.5),t1_WC_DA(7.5),t1_ST_DA(7.5),t2_NC_DA(7.5),t2_WC_DA(7.5),t2_ST_DA(7.5),t3_NC_DA(7.
5),t4_NC_DA(7.5),t5_NC_DA(7.5), PRINTO=2
PRINT                                       CSV=T,                                      LIST=
2,t1_NC_CP(7.5),t1_WC_CP(7.5),t1_ST_CP(7.5),t2_NC_CP(7.5),t2_WC_CP(7.5),t2_ST_CP(7.5),t3_NC_CP(7.
5),t4_NC_CP(7.5),t5_NC_CP(7.5), PRINTO=2
PRINT                                       CSV=T,                                      LIST=
3,t1_NC_CX(7.5),t1_WC_CX(7.5),t1_ST_CX(7.5),t2_NC_CX(7.5),t2_WC_CX(7.5),t2_ST_CX(7.5),t3_NC_CX(7.
5),t4_NC_CX(7.5),t5_NC_CX(7.5), PRINTO=2
```

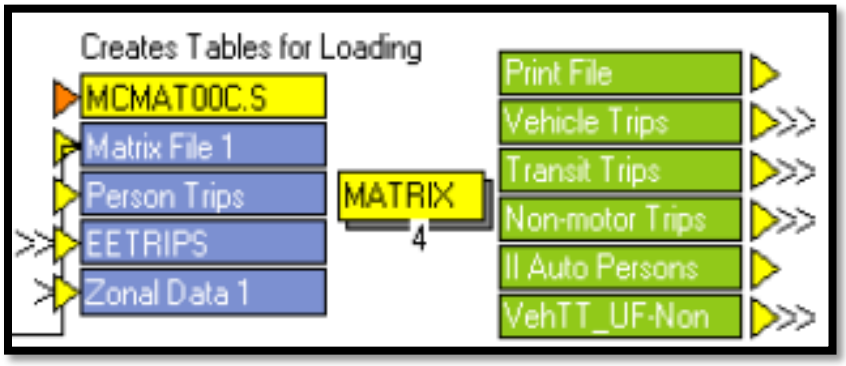
```
PRINT                                                CSV=T,                                                LIST=
4,t1_NC_WB(7.5),t1_WC_WB(7.5),t1_ST_WB(7.5),t2_NC_WB(7.5),t2_WC_WB(7.5),t2_ST_WB(7.5),t3_NC_WB(7.
5),t4_NC_WB(7.5),t5_NC_WB(7.5), PRINTO=2
PRINT                                                CSV=T,                                                LIST=
5,t1_NC_WX(7.5),t1_WC_WX(7.5),t1_ST_WX(7.5),t2_NC_WX(7.5),t2_WC_WX(7.5),t2_ST_WX(7.5),t3_NC_WX(7.
5),t4_NC_WX(7.5),t5_NC_WX(7.5), PRINTO=2
PRINT                                                CSV=T,                                                LIST=
6,t1_NC_BA(7.5),t1_WC_BA(7.5),t1_ST_BA(7.5),t2_NC_BA(7.5),t2_WC_BA(7.5),t2_ST_BA(7.5),t3_NC_BA(7.
5),t4_NC_BA(7.5),t5_NC_BA(7.5), PRINTO=2
PRINT                                                CSV=T,                                                LIST=
7,t1_NC_WK(7.5),t1_WC_WK(7.5),t1_ST_WK(7.5),t2_NC_WK(7.5),t2_WC_WK(7.5),t2_ST_WK(7.5),t3_NC_WK(7.
5),t4_NC_WK(7.5),t5_NC_WK(7.5), PRINTO=2
PRINT                                                CSV=T,                                                LIST=
8,t1_NC_BK(7.5),t1_WC_BK(7.5),t1_ST_BK(7.5),t2_NC_BK(7.5),t2_WC_BK(7.5),t2_ST_BK(7.5),t3_NC_BK(7.
5),t4_NC_BK(7.5),t5_NC_BK(7.5), PRINTO=2

; Shares
PRINT CSV=T, LIST= 'MODAL SHARES', PRINTO=2
PRINT                                                CSV=T,                                                LIST=
1,MW401(7.5),MW411(7.5),MW421(7.5),MW431(7.5),MW441(7.5),MW451(7.5),MW171(7.5),MW181(7.5),0(7.5)
, PRINTO=2
PRINT                                                CSV=T,                                                LIST=
2,MW402(7.5),MW412(7.5),MW422(7.5),MW432(7.5),MW442(7.5),MW452(7.5),MW172(7.5),MW182(7.5),0(7.5)
, PRINTO=2
PRINT                                                CSV=T,                                                LIST=
3,MW403(7.5),MW413(7.5),MW423(7.5),MW433(7.5),MW443(7.5),MW453(7.5),MW173(7.5),MW183(7.5),0(7.5)
, PRINTO=2
PRINT                                                CSV=T,                                                LIST=
4,MW404(7.5),MW414(7.5),MW424(7.5),MW434(7.5),MW444(7.5),MW454(7.5),MW174(7.5),MW184(7.5),MW191(7
.5)   , PRINTO=2
PRINT                                                CSV=T,                                                LIST=
5,MW405(7.5),MW415(7.5),MW425(7.5),MW435(7.5),MW445(7.5),MW455(7.5),MW175(7.5),MW185(7.5),0(7.5)
, PRINTO=2
PRINT                                                CSV=T,                                                LIST=
6,MW406(7.5),MW416(7.5),MW426(7.5),MW436(7.5),MW446(7.5),MW456(7.5),MW176(7.5),MW186(7.5),0(7.5)
, PRINTO=2
PRINT                                                CSV=T,                                                LIST=
7,MW407(7.5),MW417(7.5),MW427(7.5),MW437(7.5),MW447(7.5),MW457(7.5),MW177(7.5),MW187(7.5),MW192(7
.5)   , PRINTO=2
PRINT                                                CSV=T,                                                LIST=
8,MW408(7.5),MW418(7.5),MW428(7.5),MW438(7.5),MW448(7.5),MW458(7.5),MW178(7.5),MW188(7.5),MW193(7
.5)   , PRINTO=2
PRINT CSV=T, LIST= 'T',mx1,mx2,mx3,mx4,mx5,mx6,mx7,mx8,mx9   , PRINTO=2

; print INPUT modal constants
PRINT CSV=T, LIST= 'INPUT CONSTANTS', PRINTO=2
PRINT                                                CSV=T,                                                LIST=
1.0,K1_NC_DA(10.5),K1_WC_DA(10.5),K1_ST_DA(10.5),K2_NC_DA(10.5),K2_WC_DA(10.5),K2_ST_DA(10.5),K3_
NC_DA(10.5),K4_NC_DA(10.5),K5_NC_DA(10.5), PRINTO=2
PRINT                                                CSV=T,                                                LIST=
2.0,K1_NC_CP(10.5),K1_WC_CP(10.5),K1_ST_CP(10.5),K2_NC_CP(10.5),K2_WC_CP(10.5),K2_ST_CP(10.5),K3_
NC_CP(10.5),K4_NC_CP(10.5),K5_NC_CP(10.5), PRINTO=2
PRINT                                                CSV=T,                                                LIST=
3.0,K1_NC_CX(10.5),K1_WC_CX(10.5),K1_ST_CX(10.5),K2_NC_CX(10.5),K2_WC_CX(10.5),K2_ST_CX(10.5),K3_
NC_CX(10.5),K4_NC_CX(10.5),K5_NC_CX(10.5), PRINTO=2
PRINT                                                CSV=T,                                                LIST=
4.0,K1_NC_WB(10.5),K1_WC_WB(10.5),K1_ST_WB(10.5),K2_NC_WB(10.5),K2_WC_WB(10.5),K2_ST_WB(10.5),K3_
NC_WB(10.5),K4_NC_WB(10.5),K5_NC_WB(10.5), PRINTO=2
PRINT                                                CSV=T,                                                LIST=
5.0,K1_NC_WX(10.5),K1_WC_WX(10.5),K1_ST_WX(10.5),K2_NC_WX(10.5),K2_WC_WX(10.5),K2_ST_WX(10.5),K3_
NC_WX(10.5),K4_NC_WX(10.5),K5_NC_WX(10.5), PRINTO=2
PRINT                                                CSV=T,                                                LIST=
6.0,K1_NC_BA(10.5),K1_WC_BA(10.5),K1_ST_BA(10.5),K2_NC_BA(10.5),K2_WC_BA(10.5),K2_ST_BA(10.5),K3_
NC_BA(10.5),K4_NC_BA(10.5),K5_NC_BA(10.5), PRINTO=2
PRINT                                                CSV=T,                                                LIST=
7.0,K1_NC_WK(10.5),K1_WC_WK(10.5),K1_ST_WK(10.5),K2_NC_WK(10.5),K2_WC_WK(10.5),K2_ST_WK(10.5),K3_
NC_WK(10.5),K4_NC_WK(10.5),K5_NC_WK(10.5), PRINTO=2
PRINT                                                CSV=T,                                                LIST=
8.0,K1_NC_BK(10.5),K1_WC_BK(10.5),K1_ST_BK(10.5),K2_NC_BK(10.5),K2_WC_BK(10.5),K2_ST_BK(10.5),K3_
NC_BK(10.5),K4_NC_BK(10.5),K5_NC_BK(10.5), PRINTO=2
```

```
if ({MC_Cal}>1) ; Calibrate?
; -- Revised constants
D=+1.0    ; Dampening factor
; HBW No+Car: DA(1) and BA(6) not in set, No WX present so omit from calibration
R1_NC_DA=0
/*
NESTMOTOR    = 1.0
NESTTRANSIT  = 1.0
NESTNONMOTOR = 1.0
*/

LCP=Ln(MW402/t1_NC_CP)*NESTMOTOR
LCX=Ln(MW403/t1_NC_CX)*NESTMOTOR
LWB=Ln(MW404/t1_NC_WB)*NESTTRANSIT
LWK=Ln(MW407/t1_NC_WK)*NESTNONMOTOR
LBK=Ln(MW408/t1_NC_BK)*NESTNONMOTOR

R1_NC_CP=K1_NC_CP
R1_NC_CX=K1_NC_CX+D*(+LCP-LCX)
R1_NC_WB=K1_NC_WB+D*(+LCP-LWB)
R1_NC_WX=R1_NC_WB ;  express same as local
R1_NC_BA=K1_NC_BA
R1_NC_WK=K1_NC_WK+D*(+LCP-LWK)
R1_NC_BK=K1_NC_BK+D*(+LCP-LBK)

; HBW With+Car: No WX present so omit from calibration
;R1_WC_DA=K1_WC_DA+D*(-
Ln(MW411/t1_WC_DA)+Ln(MW412/t1_WC_CP)+Ln(MW413/t1_WC_CX)+Ln(MW414/t1_WC_WB)+Ln(MW416/t1_WC_BA)+Ln
(MW417/t1_WC_WK)+Ln(MW418/t1_WC_BK)
R1_WC_DA=0

LDA=Ln(MW411/t1_WC_DA)*NESTMOTOR
LCP=Ln(MW412/t1_WC_CP)*NESTMOTOR
LCX=Ln(MW413/t1_WC_CX)*NESTMOTOR
LWB=Ln(MW414/t1_WC_WB)*NESTTRANSIT
LBA=Ln(MW416/t1_WC_BA)*NESTTRANSIT
LWK=Ln(MW417/t1_WC_WK)*NESTNONMOTOR
LBK=Ln(MW418/t1_WC_BK)*NESTNONMOTOR

R1_WC_CP=K1_WC_CP+D*(+LDA-LCP)
R1_WC_CX=K1_WC_CX+D*(+LDA-LCX)
R1_WC_WB=K1_WC_WB+D*(+LDA-LWB)
R1_WC_WX=R1_WC_WB ;  express same as local
R1_WC_BA=K1_WC_BA+D*(+LDA-LBA)
R1_WC_WK=K1_WC_WK+D*(+LDA-LWK)
R1_WC_BK=K1_WC_BK+D*(+LDA-LBK)

; HBW Student: No WX present so omit from calibration
;R1_ST_DA=K1_ST_DA+D*(-
Ln(MW421/t1_ST_DA)+Ln(MW422/t1_ST_CP)+Ln(MW423/t1_ST_CX)+Ln(MW424/t1_ST_WB)+Ln(MW426/t1_ST_BA)+Ln
(MW427/t1_ST_WK)+Ln(MW428/t1_ST_BK)
R1_ST_DA=0
LDA=Ln(MW421/t1_ST_DA)*NESTMOTOR
LCP=Ln(MW422/t1_ST_CP)*NESTMOTOR
LCX=Ln(MW423/t1_ST_CX)*NESTMOTOR
LWB=Ln(MW424/t1_ST_WB)*NESTTRANSIT
LBA=Ln(MW426/t1_ST_BA)*NESTTRANSIT
LWK=Ln(MW427/t1_ST_WK)*NESTNONMOTOR
LBK=Ln(MW428/t1_ST_BK)*NESTNONMOTOR

R1_ST_CP=K1_ST_CP+D*(+LDA-LCP)
R1_ST_CX=K1_ST_CX+D*(+LDA-LCX)
R1_ST_WB=K1_ST_WB+D*(+LDA-LWB)
R1_ST_WX=R1_ST_WB ;  express same as local
R1_ST_BA=K1_ST_BA+D*(+LDA-LBA)
R1_ST_WK=K1_ST_WK+D*(+LDA-LWK)
R1_ST_BK=K1_ST_BK+D*(+LDA-LBK)

; HBO No+Car: DA(1) and BA(6) not in set, No WX present so omit from calibration
;R2_NC_DA=K2_NC_DA
R2_NC_DA=0
```

```
LCP=Ln(MW432/t2_NC_CP)*NESTMOTOR
LCX=Ln(MW433/t2_NC_CX)*NESTMOTOR
LWB=Ln(MW434/t2_NC_WB)*NESTTRANSIT
LWK=Ln(MW437/t2_NC_WK)*NESTNONMOTOR
LBK=Ln(MW438/t2_NC_BK)*NESTNONMOTOR

R2_NC_CP=K2_NC_CP
R2_NC_CX=K2_NC_CX+D*(+LCP-LCX)
R2_NC_WB=K2_NC_WB+D*(+LCP-LWB)

R2_NC_WX=R2_NC_WB ;  express same as local
R2_NC_BA=K2_NC_BA
R2_NC_WK=K2_NC_WK+D*(+LCP-LWK)
R2_NC_BK=K2_NC_BK+D*(+LCP-LBK)

; HBO With+Car: No WX present so omit from calibration
;R2_WC_DA=K2_WC_DA-
Ln(MW441/t2_WC_DA)+Ln(MW442/t2_WC_CP)+Ln(MW443/t2_WC_CX)+Ln(MW444/t2_WC_WB)+Ln(MW446/t2_WC_BA)+Ln
(MW447/t2_WC_WK)+Ln(MW448/t2_WC_BK)
R2_WC_DA=0
LDA=Ln(MW441/t2_WC_DA)*NESTMOTOR
LCP=Ln(MW442/t2_WC_CP)*NESTMOTOR
LCX=Ln(MW443/t2_WC_CX)*NESTMOTOR
LWB=Ln(MW444/t2_WC_WB)*NESTTRANSIT
LBA=Ln(MW446/t2_WC_BA)*NESTTRANSIT
LWK=Ln(MW447/t2_WC_WK)*NESTNONMOTOR
LBK=Ln(MW448/t2_WC_BK)*NESTNONMOTOR

R2_WC_CP=K2_WC_CP+D*(+LDA-LCP)
R2_WC_CX=K2_WC_CX+D*(+LDA-LCX)
R2_WC_WB=K2_WC_WB+D*(+LDA-LWB)
R2_WC_WX=R2_WC_WB ;  express same as local
R2_WC_BA=K2_WC_BA+D*(+LDA-LBA)
R2_WC_WK=K2_WC_WK+D*(+LDA-LWK)
R2_WC_BK=K2_WC_BK+D*(+LDA-LBK)

; HBO Student: No WX present so omit from calibration
;R2_ST_DA=K2_ST_DA-
Ln(MW451/t2_ST_DA)+Ln(MW452/t2_ST_CP)+Ln(MW453/t2_ST_CX)+Ln(MW454/t2_ST_WB)+Ln(MW456/t2_ST_BA)+Ln
(MW457/t2_ST_WK)+Ln(MW458/t2_ST_BK)
R2_ST_DA=0
LDA=Ln(MW451/t2_ST_DA)*NESTMOTOR
LCP=Ln(MW452/t2_ST_CP)*NESTMOTOR
LCX=Ln(MW453/t2_ST_CX)*NESTMOTOR
LWB=Ln(MW454/t2_ST_WB)*NESTTRANSIT
LBA=Ln(MW456/t2_ST_BA)*NESTTRANSIT
LWK=Ln(MW457/t2_ST_WK)*NESTNONMOTOR
LBK=Ln(MW458/t2_ST_BK)*NESTNONMOTOR

R2_ST_CP=K2_ST_CP+D*(+LDA-LCP)
R2_ST_CX=K2_ST_CX+D*(+LDA-LCX)
R2_ST_WB=K2_ST_WB+D*(+LDA-LWB)
R2_ST_WX=R2_ST_WB ;  express same as local
R2_ST_BA=K2_ST_BA+D*(+LDA-LBA)
R2_ST_WK=K2_ST_WK+D*(+LDA-LWK)
R2_ST_BK=K2_ST_BK+D*(+LDA-LBK)

; NHB: No WX present so omit from calibration
;R3_NC_DA=K3_NC_DA-
Ln(MW171/t3_NC_DA)+Ln(MW172/t3_NC_CP)+Ln(MW173/t3_NC_CX)+Ln(MW174/t3_NC_WB)+Ln(MW176/t3_NC_BA)+Ln
(MW177/t3_NC_WK)+Ln(MW178/t3_NC_BK)
R3_NC_DA=0
LDA=Ln(MW171/t3_NC_DA)*NESTMOTOR
LCP=Ln(MW172/t3_NC_CP)*NESTMOTOR
LCX=Ln(MW173/t3_NC_CX)*NESTMOTOR
LWB=Ln(MW174/t3_NC_WB)*NESTTRANSIT
LBA=Ln(MW176/t3_NC_BA)*NESTTRANSIT
LWK=Ln(MW177/t3_NC_WK)*NESTNONMOTOR
LBK=Ln(MW178/t3_NC_BK)*NESTNONMOTOR
```

```
R3_NC_CP=K3_NC_CP+D*(+LDA-LCP)
R3_NC_CX=K3_NC_CX+D*(+LDA-LCX)
R3_NC_WB=K3_NC_WB+D*(+LDA-LWB)
R3_NC_WX=R3_NC_WB ;  express same as local
R3_NC_BA=K3_NC_BA+D*(+LDA-LBA)
R3_NC_WK=K3_NC_WK+D*(+LDA-LWK)
R3_NC_BK=K3_NC_BK+D*(+LDA-LBK)

; HBU: No WX present so omit from calibration
;R4_NC_DA=K4_NC_DA-
Ln(MW181/t4_NC_DA)+Ln(MW182/t4_NC_CP)+Ln(MW183/t4_NC_CX)+Ln(MW184/t4_NC_WB)+Ln(MW186/t4_NC_BA)+Ln
(MW187/t4_NC_WK)+Ln(MW188/t4_NC_BK)
R4_NC_DA=0
LDA=Ln(MW181/t4_NC_DA)*NESTMOTOR
LCP=Ln(MW182/t4_NC_CP)*NESTMOTOR
LCX=Ln(MW183/t4_NC_CX)*NESTMOTOR
LWB=Ln(MW184/t4_NC_WB)*NESTTRANSIT
LBA=Ln(MW186/t4_NC_BA)*NESTTRANSIT
LWK=Ln(MW187/t4_NC_WK)*NESTNONMOTOR
LBK=Ln(MW188/t4_NC_BK)*NESTNONMOTOR

R4_NC_CP=K4_NC_CP+D*(+LDA-LCP)
R4_NC_CX=K4_NC_CX+D*(+LDA-LCX)
R4_NC_WB=K4_NC_WB+D*(+LDA-LWB)
R4_NC_WX=R4_NC_WB ;  express same as local
R4_NC_BA=K4_NC_BA+D*(+LDA-LBA)
R4_NC_WK=K4_NC_WK+D*(+LDA-LWK)
R4_NC_BK=K4_NC_BK+D*(+LDA-LBK)

; DORM: AUTO, PNR and WX NOT IN CHOICE SET. No WX present so omit from calibration
R5_NC_DA=0
LWB=Ln(MW191/t5_NC_WB)*NESTTRANSIT
LWK=Ln(MW192/t5_NC_WK)*NESTNONMOTOR
LBK=Ln(MW193/t5_NC_BK)*NESTNONMOTOR

R5_NC_CP=0
R5_NC_CX=0
R5_NC_WB=K5_NC_WB
R5_NC_WX=0
R5_NC_BA=0
R5_NC_WK=K5_NC_WK+D*(+LWB-LWK)
R5_NC_BK=K5_NC_BK+D*(+LWB-LBK)

; print REVISED modal constants
PRINT CSV=T, LIST= 'REVISED CONSTANTS', PRINTO=2
PRINT                                          CSV=T,                                                  LIST=
1.0,R1_NC_DA(10.5),R1_WC_DA(10.5),R1_ST_DA(10.5),R2_NC_DA(10.5),R2_WC_DA(10.5),R2_ST_DA(10.5),R3_
NC_DA(10.5),R4_NC_DA(10.5),R5_NC_DA(10.5), PRINTO=2
PRINT                                          CSV=T,                                                  LIST=
2.0,R1_NC_CP(10.5),R1_WC_CP(10.5),R1_ST_CP(10.5),R2_NC_CP(10.5),R2_WC_CP(10.5),R2_ST_CP(10.5),R3_
NC_CP(10.5),R4_NC_CP(10.5),R5_NC_CP(10.5), PRINTO=2
PRINT                                          CSV=T,                                                  LIST=
3.0,R1_NC_CX(10.5),R1_WC_CX(10.5),R1_ST_CX(10.5),R2_NC_CX(10.5),R2_WC_CX(10.5),R2_ST_CX(10.5),R3_
NC_CX(10.5),R4_NC_CX(10.5),R5_NC_CX(10.5), PRINTO=2
PRINT                                          CSV=T,                                                  LIST=
4.0,R1_NC_WB(10.5),R1_WC_WB(10.5),R1_ST_WB(10.5),R2_NC_WB(10.5),R2_WC_WB(10.5),R2_ST_WB(10.5),R3_
NC_WB(10.5),R4_NC_WB(10.5),R5_NC_WB(10.5), PRINTO=2
PRINT                                          CSV=T,                                                  LIST=
5.0,R1_NC_WX(10.5),R1_WC_WX(10.5),R1_ST_WX(10.5),R2_NC_WX(10.5),R2_WC_WX(10.5),R2_ST_WX(10.5),R3_
NC_WX(10.5),R4_NC_WX(10.5),R5_NC_WX(10.5), PRINTO=2
PRINT                                          CSV=T,                                                  LIST=
6.0,R1_NC_BA(10.5),R1_WC_BA(10.5),R1_ST_BA(10.5),R2_NC_BA(10.5),R2_WC_BA(10.5),R2_ST_BA(10.5),R3_
NC_BA(10.5),R4_NC_BA(10.5),R5_NC_BA(10.5), PRINTO=2
PRINT                                          CSV=T,                                                  LIST=
7.0,R1_NC_WK(10.5),R1_WC_WK(10.5),R1_ST_WK(10.5),R2_NC_WK(10.5),R2_WC_WK(10.5),R2_ST_WK(10.5),R3_
NC_WK(10.5),R4_NC_WK(10.5),R5_NC_WK(10.5), PRINTO=2
PRINT                                          CSV=T,                                                  LIST=
8.0,R1_NC_BK(10.5),R1_WC_BK(10.5),R1_ST_BK(10.5),R2_NC_BK(10.5),R2_WC_BK(10.5),R2_ST_BK(10.5),R3_
NC_BK(10.5),R4_NC_BK(10.5),R5_NC_BK(10.5), PRINTO=2

; print REVISED modal constants
```

```
PRINT                                          CSV=T,                                          LIST=
1.0,R1_NC_DA(10.5),R1_WC_DA(10.5),R1_ST_DA(10.5),R2_NC_DA(10.5),R2_WC_DA(10.5),R2_ST_DA(10.5),R3_
NC_DA(10.5),R4_NC_DA(10.5),R5_NC_DA(10.5), PRINTO=3
PRINT                                          CSV=T,                                          LIST=
2.0,R1_NC_CP(10.5),R1_WC_CP(10.5),R1_ST_CP(10.5),R2_NC_CP(10.5),R2_WC_CP(10.5),R2_ST_CP(10.5),R3_
NC_CP(10.5),R4_NC_CP(10.5),R5_NC_CP(10.5), PRINTO=3
PRINT                                          CSV=T,                                          LIST=
3.0,R1_NC_CX(10.5),R1_WC_CX(10.5),R1_ST_CX(10.5),R2_NC_CX(10.5),R2_WC_CX(10.5),R2_ST_CX(10.5),R3_
NC_CX(10.5),R4_NC_CX(10.5),R5_NC_CX(10.5), PRINTO=3
PRINT                                          CSV=T,                                          LIST=
4.0,R1_NC_WB(10.5),R1_WC_WB(10.5),R1_ST_WB(10.5),R2_NC_WB(10.5),R2_WC_WB(10.5),R2_ST_WB(10.5),R3_
NC_WB(10.5),R4_NC_WB(10.5),R5_NC_WB(10.5), PRINTO=3
PRINT                                          CSV=T,                                          LIST=
5.0,R1_NC_WX(10.5),R1_WC_WX(10.5),R1_ST_WX(10.5),R2_NC_WX(10.5),R2_WC_WX(10.5),R2_ST_WX(10.5),R3_
NC_WX(10.5),R4_NC_WX(10.5),R5_NC_WX(10.5), PRINTO=3
PRINT                                          CSV=T,                                          LIST=
6.0,R1_NC_BA(10.5),R1_WC_BA(10.5),R1_ST_BA(10.5),R2_NC_BA(10.5),R2_WC_BA(10.5),R2_ST_BA(10.5),R3_
NC_BA(10.5),R4_NC_BA(10.5),R5_NC_BA(10.5), PRINTO=3
PRINT                                          CSV=T,                                          LIST=
7.0,R1_NC_WK(10.5),R1_WC_WK(10.5),R1_ST_WK(10.5),R2_NC_WK(10.5),R2_WC_WK(10.5),R2_ST_WK(10.5),R3_
NC_WK(10.5),R4_NC_WK(10.5),R5_NC_WK(10.5), PRINTO=3
PRINT                                          CSV=T,                                          LIST=
8.0,R1_NC_BK(10.5),R1_WC_BK(10.5),R1_ST_BK(10.5),R2_NC_BK(10.5),R2_WC_BK(10.5),R2_ST_BK(10.5),R3_
NC_BK(10.5),R4_NC_BK(10.5),R5_NC_BK(10.5), PRINTO=3

endif ; MC_Cal

ENDIF
ENDRUN
```

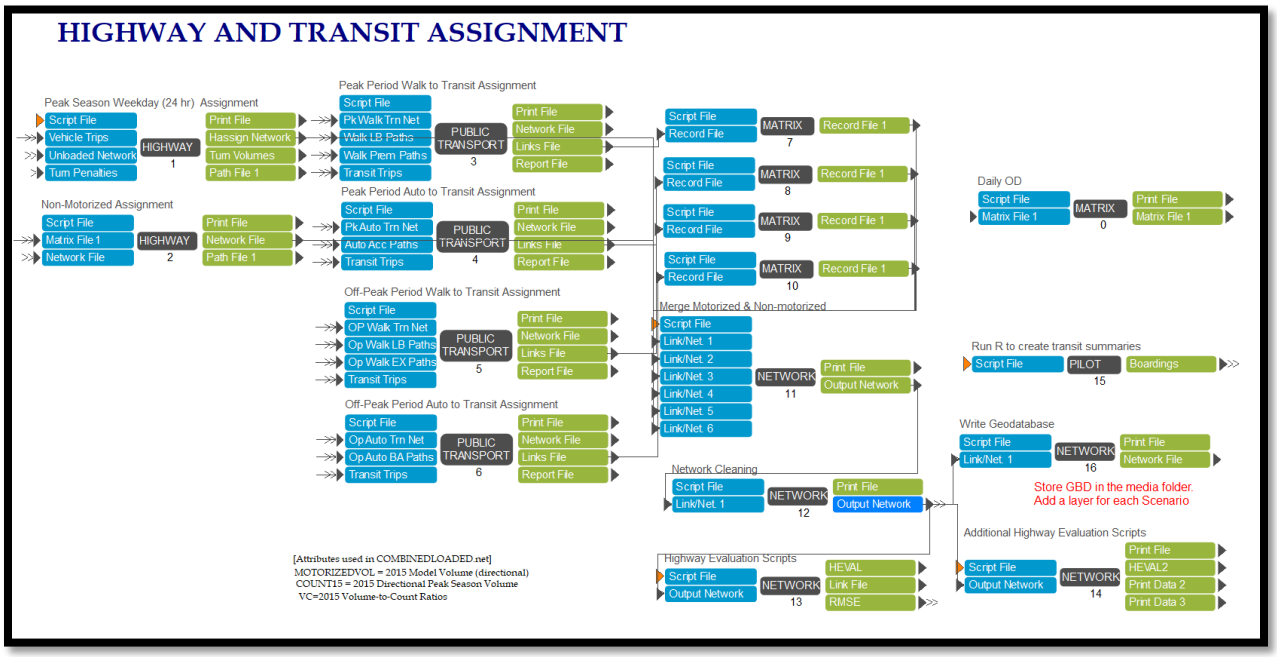


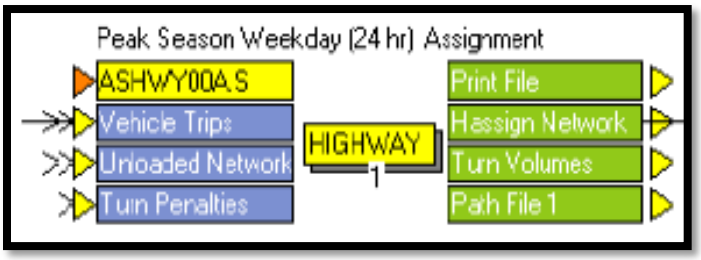
MCMAT00C.S

```
; Do not change filenames or add or remove FILEI/FILEO statements using an editor. Use
Cube/Application Manager.
RUN PGM=MATRIX PRNFILE="{SCENARIO_DIR}\output\FINALTABLES.PRN" MSG='Creates Tables for Loading'
FILEI MATI[3] = "{SCENARIO_DIR}\output\EETAB.MAT"
FILEI MATI[2] = "{SCENARIO_DIR}\output\PTRIPS.MAT"
FILEO MATO[5] = "{SCENARIO_DIR}\output\VEHSBYUF_NON.MAT",
MO=41,42,51,52,61,62,
 NAME=UF_light,UF_heavy,Non_light,Non_heavy,SelZones_light,SelZones_heavy, DEC=6*S
FILEI ZDATI[1] = "{SCENARIO_DIR}\input\ZoneData{YEAR}.DBF", Z=TAZ_20{year}
FILEO MATO[4] = "{SCENARIO_DIR}\output\IIAUTOPERSONS.MAT",
 MO=31, NAME=iiAutoPersons, DEC=5*S
FILEO MATO[3] = "{SCENARIO_DIR}\output\NONMOTOR.MAT",
 MO=21-22, NAME=WALK,BIKE, DEC=2*S
FILEO MATO[2] = "{SCENARIO_DIR}\output\TRANSIT.MAT",
 MO=11-16, NAME=PKWALKLOCAL,PKWALKPREM,PKAUTOBA,OPWALKLOCAL, OPWALKPREM,OPAUTOBA, DEC=6*S
FILEO MATO[1] = "{SCENARIO_DIR}\output\VEHICLES.MAT", MO=1-5,
 NAME=DRIVEALONE,CARPOOL,LIGHTTRUCK,HEAVYTRUCK,EETRIPS, DEC=5*S
FILEI MATI[1] = "{SCENARIO_DIR}\output\MODEOUT.MAT"
PARAMETERS ZONEMSG=100
; DRIVE ALONE
MW[1]=((MI.1.HBWDA+MI.1.HBWDA.T)/1.0+
       (MI.1.HBODA+MI.1.HBODA.T)/1.0+
```

```
       (MI.1.NHBDA+MI.1.NHBDA.T)/1.0+
       (MI.1.HBUDA+MI.1.HBUDA.T)/1.0+
       (MI.2.SOVIE+MI.2.SOVIE.T)/1.0)*0.50
; CARPOOL
MW[2]=((MI.1.HBWCP+MI.1.HBWCP.T)/2.0+
       (MI.1.HBOCP+MI.1.HBOCP.T)/2.0+
       (MI.1.NHBCP+MI.1.NHBCP.T)/2.0+
       (MI.1.HBUCP+MI.1.HBUCP.T)/2.0+
       (MI.1.HBWCX+MI.1.HBWCX.T)/{HBW3P}+
       (MI.1.HBOCX+MI.1.HBOCX.T)/{HBO3P}+
       (MI.1.NHBCX+MI.1.NHBCX.T)/{NHB3P}+
       (MI.1.HBUCX+MI.1.HBUCX.T)/{HBW3P}+ ; ASSUME 3+ occ same as work
       (MI.2.HOVIE+MI.2.HOVIE.T)/1.0)*0.50
; LIGHT DUTY TRUCKS
MW[3]=((MI.2.TRUCK4+MI.2.TRUCK4.T)+(MI.2.TRUCKLDIE+MI.2.TRUCKLDIE.T)+(MI.2.TRUCKSU+MI.2.TRUCKSU.T
))*0.50
; HEAVY DUTY TRUCKS
MW[4]=((MI.2.TRUCKTRLR+MI.2.TRUCKTRLR.T)+(MI.2.TRUCKHDIE+MI.2.TRUCKHDIE.T))*0.50+MI.3.EETTRIPS   ;
0.5 added by KDK; EETTRIPS added by Srin
; EETRIPS
MW[5]=MI.3.1
; TRANSIT, PEAK PERIOD
MW[011]=MI.1.HBWWB
MW[012]=MI.1.HBWWX
MW[013]=MI.1.HBWBA
; TRANSIT, OFF-PEAK PERIOD
MW[014]=MI.1.HBOWB+MI.1.NHBWB+MI.1.HBUWB+MI.1.HDORMUWB
MW[015]=MI.1.HBOWX+MI.1.NHBWX+MI.1.HBUWX
MW[016]=MI.1.HBOBA+MI.1.NHBBA+MI.1.HBUBA
; NON-MOTORIZED
MW[021]=MI.1.HBWWK+MI.1.HBOWK+MI.1.NHBWK+MI.1.HBUWK+MI.1.HDORMUWK
MW[022]=MI.1.HBWBK+MI.1.HBOBK+MI.1.NHBBK+MI.1.HBUBK+MI.1.HDORMUBK
; Internal Auto Persons
MW[31]= MI.1.HBWDA+        MI.1.HBODA+        MI.1.NHBDA+
        MI.1.HBUDA+        MI.1.HBWCP+        MI.1.HBOCP+
        MI.1.NHBCP+        MI.1.HBUCP+        MI.1.HBWCX+
        MI.1.HBOCX+        MI.1.NHBCX+        MI.1.HBUCX

; Select Zone Vehicle Trips
  JLOOP
   if(i=1 & j=1) MW[011]=MW[011]+0.01
   if(i=1 & j=1) MW[012]=MW[012]+0.01
   if(i=1 & j=1) MW[013]=MW[013]+0.01
   if(i=1 & j=1) MW[014]=MW[014]+0.01
   if(i=1 & j=1) MW[015]=MW[015]+0.01
   if(i=1 & j=1) MW[016]=MW[016]+0.01

   if((ZI.1.SELECTZONE[J]=1) || (ZI.1.SELECTZONE[I]=1)) ; Select Zones
     mw[61] = mw[1] + mw[2] + mw[3] +mw[5] ; Select Zones light Vehicles
     mw[62] = mw[4]                        ; Select Zones heavy vehicles
   endif

; UF vs Non-UF Vehicle Trips
   if((ZI.1.UFZONES[J]=1) || (ZI.1.UFZONES[I]=1)) ; UF related
     mw[41] = mw[1] + mw[2] + mw[3] +mw[5] ; UF light Vehicles
     mw[42] = mw[4]                        ; UF heavy vehicles
   else
     mw[51] = mw[1] + mw[2] + mw[3] +mw[5] ; non-UF light Vehicles
     mw[52] = mw[4]                        ; non-UF heavy vehicles
   endif
  ENDJLOOP
ENDRUN
```

Assignment Step



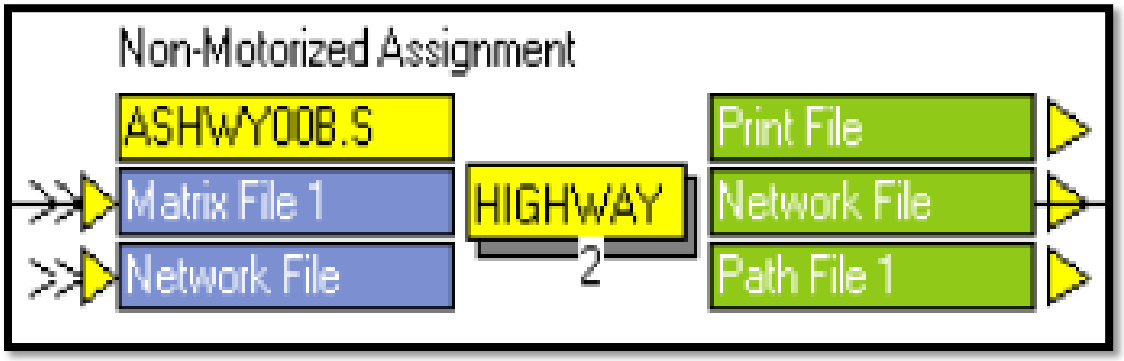


ASHWY00A.S

```
; Do not change filenames or add or remove FILEI/FILEO statements using an editor. Use
Cube/Application Manager.
RUN PGM=HIGHWAY PRNFILE="{SCENARIO_DIR}\output\HASSIGN.PRN" MSG='Peak Season Weekday (24 hr)
Assignment'
FILEI NETI = "{SCENARIO_DIR}\output\UNLOADED.NET"
FILEO TURNVOLO = "{SCENARIO_DIR}\output\TURNVOL.BIN",
FORMAT=DBF
TURNS N=1-999999, T=TURN[1] + TURN[2] +TURN[3] +TURN[4]
FILEI MATI[1] = "{SCENARIO_DIR}\output\VEHSBYUF_NON.MAT"
FILEO NETO = "{SCENARIO_DIR}\output\HASSIGN.NET"
FILEO PATHO[1] = "{SCENARIO_DIR}\output\AUTOS.PTH"
FILEI TURNPENI = "{SCENARIO_DIR}\input\TCARDS.PEN"
;parameters zonemsg=100
PARAMETERS,
        ZONES={zonesa},
        MAXITERS=100,
        COMBINE=EQUI,ENHANCE=2, BIFWVERSION=602,
        RELATIVEGAP=0.0001
PROCESS PHASE=LINKREAD

; Establish LINKCLASS for capacity restraint
      ;assign all other roadways with any FTYPE
 IF (li.FTYPE=51-59)
 LINKCLASS = 1 ;centroid connectors
```

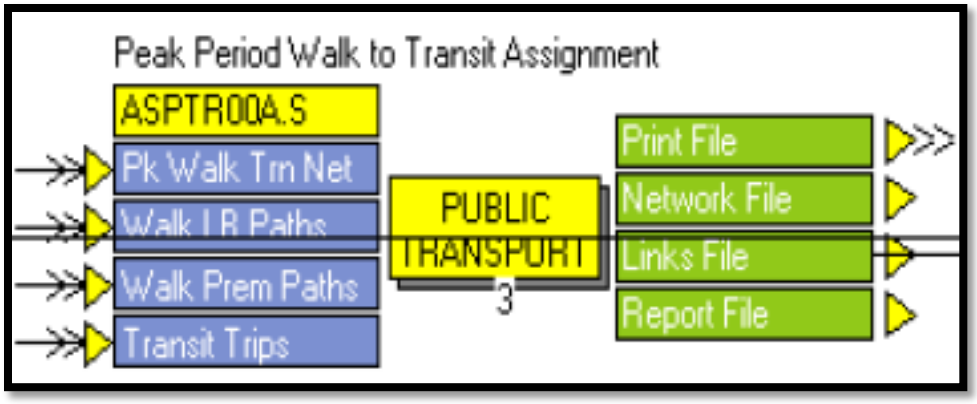
```
  ELSEIF (li.FTYPE=11-19)
  LINKCLASS = 2        ; Freeways
  ELSEIF (li.FTYPE=21-29)
  LINKCLASS = 3        ; Divided hwys
  ELSEIF (li.FTYPE=31-39)
  LINKCLASS = 5        ; undivided Hwys
  ELSEIF (li.FTYPE=41-49)
  LINKCLASS = 5         ; Collectors
  ELSEIF (li.FTYPE=61-68)
  LINKCLASS = 5       ; Ramps/oneways
  ELSEIF(  Li.FTYPE=71-79)
  LINKCLASS = 5
  ELSE
   LINKCLASS=5
ENDIF

; USE THE USER SUPPLIED ALPHA AND BETA FOR THE BPR CURVE
  ;IF (LI.BPRCOEFFICIENT=0)
     ;LW.BPRCOEFFICIENT=0.15
  ;ELSE
     ;LW.BPRCOEFFICIENT=LI.BPRCOEFFICIENT
  ;ENDIF
  ;IF (LI.BPREXPONENT=0)
     ;LW.BPREXPONENT=4.0
  ;ELSE
     ;LW.BPREXPONENT=LI.BPREXPONENT
  ;ENDIF
;BPR equations have parameter alpha (0-2), and beta (0-10)
    lw.Alpha12=0.15 ;freeways
    lw.Beta12=4
    lw.Alpha13=0.15 ;divided
    lw.Beta13=4
    lw.Alpha14=0.15  ;undivided
    lw.Beta14=4
    lw.Alpha15=0.15 ;collectors
    lw.Beta15=4
         ;ENDIF

  IF (LI.CAPACITY=0)
     LW.DAILYCAP=999999
  ELSE
      LW.DAILYCAP=(LI.CAPACITY/li.confac)*li.uroadfactor
  ENDIF
  IF (LI.TIME=0)
     LW.FFTIME=0.00001
  ELSE
     LW.FFTIME=LI.TIME
  ENDIF
C=LW.DAILYCAP
T0=LW.FFTIME
IF (LI.FTYPE=49) ADDTOGROUP=1
ENDPROCESS

PROCESS PHASE=ILOOP
  MW[1]=MI.1.UF_light
  MW[2]=MI.1.UF_heavy;*{PCE_HT}- changed trucks to vehciles, not PCE here.
  MW[3]=MI.1.Non_light
  MW[4]=MI.1.Non_heavy;*{PCE_HT}-changed trucks to vehciles, not PCE here
  MW[5]=MI.1.SelZones_light
  MW[6]=MI.1.SelZones_heavy;*{PCE_HT}-changed trucks to vehciles, not PCE here

  PATHLOAD                                                                    PATH=TIME,
VOL[1]=MW[1],VOL[2]=MW[2],VOL[3]=MW[3],VOL[4]=MW[4],VOL[5]=MW[5],VOL[6]=MW[6],    PENI=1,
EXCLUDEGROUP=1,
  PATHO=1, ALLJ=T, INCLUDECOST=F, NAME=ALLTRIPS
ENDPROCESS

PROCESS PHASE=ADJUST
  V = VOL[1] + VOL[2]*{PCE_HT}+ VOL[3] + VOL[4]*{PCE_HT}
; here convert trucks to PCE
            TC[1] = T0
```

```
            TC[2] = T0*(1 + (lw.Alpha12*(V/C)^lw.Beta12))
            TC[3] = T0*(1 + (lw.Alpha13*(V/C)^lw.Beta13))
            TC[4] = T0*(1 + (lw.Alpha14*(V/C)^lw.Beta14))
            TC[5] = T0*(1 + (lw.Alpha15*(V/C)^lw.Beta15))
ENDPROCESS

;Add Converge phase to make the assignment more stable.
PHASE=CONVERGE
     IF (ITERATION < 6) BREAK         ; do not check if fewer than 6 iterations
     IF  (  (RGAP[ITERATION]<RGAPCUTOFF)  &&  (RGAP[ITERATION-1]<RGAPCUTOFF)  &&  (RGAP[ITERATION-
2]<RGAPCUTOFF) )
       print list="Equilibrium reached at RGAP ",rgap[ITERATION](8.5), " iter= ",ITERATION(3.0)
       BALANCE=1
     ENDIF
     if(GAPCHANGEMAX(5)<=0.000001)
       print list="Iterations stopped because nothing is changing"
       print list="RGAP =",rgap[ITERATION](8.5), " iter= ",ITERATION(3.0)
       balance=1
     ENDIF
ENDPHASE
ENDRUN
```



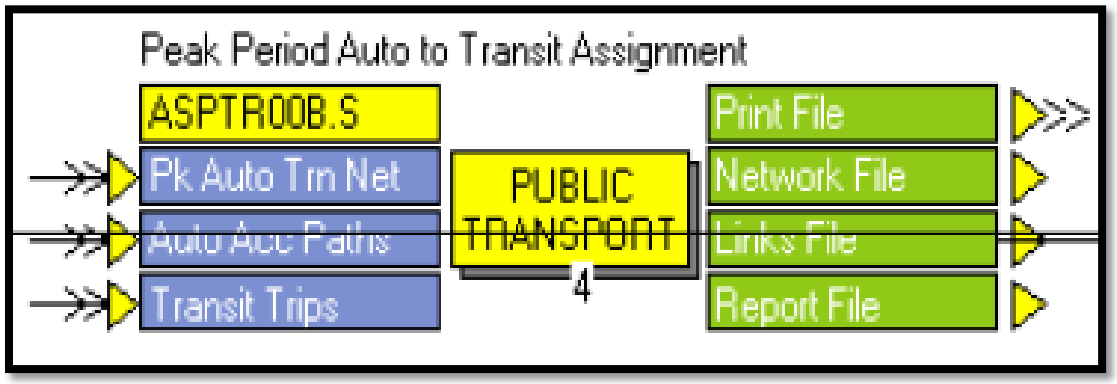
ASHWY00B.S

```
; Do not change filenames or add or remove FILEI/FILEO statements using an editor. Use
Cube/Application Manager.
RUN PGM=HIGHWAY PRNFILE="{SCENARIO_DIR}\output\ASHWY00A.PRN" MSG='Non-Motorized Assignment'
FILEI NETI = "{SCENARIO_DIR}\output\UNLOADED.NET"
FILEO PATHO[1] = "{SCENARIO_DIR}\output\NONMOTOR.PTH"
FILEO NETO = "{SCENARIO_DIR}\output\NONMOTOR.NET"
FILEI MATI[1] = "{SCENARIO_DIR}\output\NONMOTOR.MAT"

PARAMETERS MAXITERS=1

PROCESS PHASE=LINKREAD
IF (LI.FTYPE=10-19,70-99) ADDTOGROUP=1
ENDPROCESS

PROCESS PHASE=ILOOP
 PATHLOAD PATH=LI.DISTANCE, VOL[1]=MI.1.WALK, VOL[2]=MI.1.BIKE, EXCLUDEGROUP=1,
    PATHO=1,NAME='NONMOTOR',ALLJ=T,INCLUDECOSTS=F
ENDPROCESS
ENDRUN
```



ASPTR00A.S

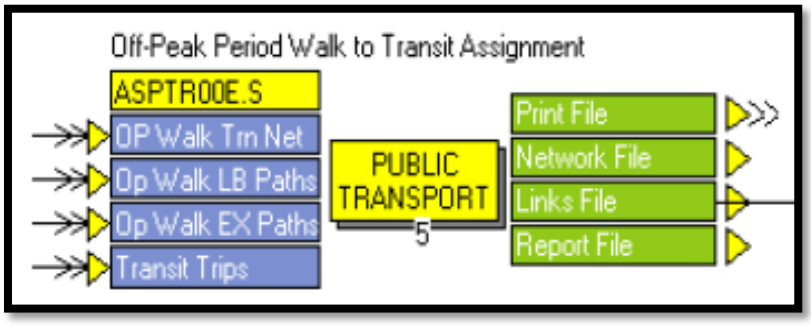
```
; Do not change filenames or add or remove FILEI/FILEO statements using an editor. Use
Cube/Application Manager.
RUN PGM=PUBLIC TRANSPORT PRNFILE="{SCENARIO_DIR}\output\TRANSIT_AMWALK.PRN" MSG='Peak Period Walk
to Transit Assignment'
FILEI ROUTEI[2] = "{SCENARIO_DIR}\output\WALKPREMAM.RTE"
FILEI ROUTEI[1] = "{SCENARIO_DIR}\output\WALKLBAM.RTE"
FILEI NETI = "{SCENARIO_DIR}\output\TNETWALKAM.NET"
FILEO LINKO = "{SCENARIO_DIR}\output\TLOD1.dbf",
    SKIP0=Y,NTLEGS=F
FILEO LINKO[2]= "{SCENARIO_DIR}\output\TLINK_OP_WALK.DBF",
    NTLEGS=N,ONOFFS=T;SKIP0=Y,NTLEGS=F
FILEI MATI[1] = "{SCENARIO_DIR}\output\TRANSIT.MAT"
FILEO REPORTO = "{SCENARIO_DIR}\output\ASPTR00B.PRN"
FILEO NETO = "{SCENARIO_DIR}\output\TLOADAM1.NET"
PARAMETERS USERCLASSES=1,2,
           TRIPSIJ[1]=(MI.1.PKWALKLOCAL),
           TRIPSIJ[2]=(MI.1.PKWALKPREM),
           NOROUTEERRS=9999999,
           NOROUTEMSGS=0
;Selection of Loading Reports
REPORT LINES=T; LINEVOLS=T STOPSONLY=T
PAGEHEIGHT=32767
ENDRUN
```



ASPTR00B.S

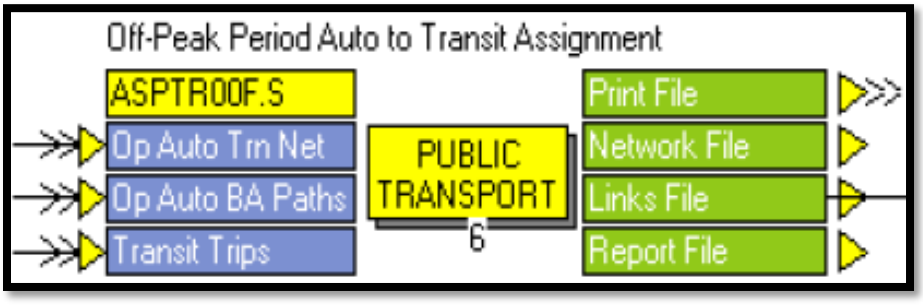
```
; Do not change filenames or add or remove FILEI/FILEO statements using an editor. Use
Cube/Application Manager.
RUN PGM=PUBLIC TRANSPORT PRNFILE="{SCENARIO_DIR}\output\TRANSIT_AMAUTO.PRN" MSG='Peak Period Auto
to Transit Assignment'
FILEI NETI = "{SCENARIO_DIR}\output\TNETAUTOAM.NET"
FILEO LINKO = "{SCENARIO_DIR}\output\TLOD2.DBF",
    SKIP0=Y,NTLEGS=F
FILEO LINKO[2] = "{SCENARIO_DIR}\output\TLINK_OP_AUTO.DBF",
```

```
    NTLEGS=N,ONOFFS=T;SKIP0=Y,NTLEGS=F
FILEO REPORTO = "{SCENARIO_DIR}\output\ASPTR00D.PRN"
FILEO NETO = "{SCENARIO_DIR}\output\TLOADAM2.NET"
FILEI MATI[1] = "{SCENARIO_DIR}\output\TRANSIT.MAT"
FILEI ROUTEI[1] = "{SCENARIO_DIR}\output\AUTOALLAM.RTE"
PARAMETERS USERCLASSES=1,
           TRIPSIJ=(MI.1.PKAUTOBA),
           NOROUTEERRS=999999,
           NOROUTEMSGS=0
REPORT LINES=T ;LINEVOLS=T
PAGEHEIGHT=32767
ENDRUN
```



ASPTR00E.S

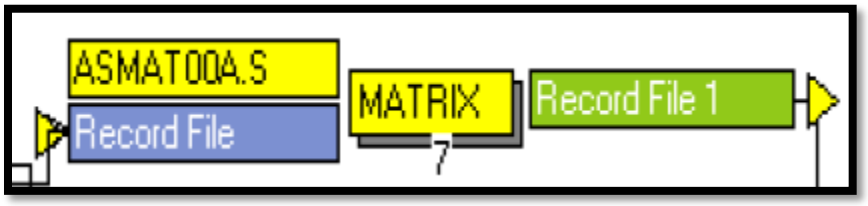
```
; Do not change filenames or add or remove FILEI/FILEO statements using an editor. Use
Cube/Application Manager.
RUN PGM=PUBLIC TRANSPORT PRNFILE="{SCENARIO_DIR}\output\TRANSIT_MDWALK.PRN" MSG='Off-Peak Period
Walk to Transit Assignment'
FILEI ROUTEI[2] = "{SCENARIO_DIR}\output\WALKPREMMD.RTE"
FILEI NETI = "{SCENARIO_DIR}\output\TNETWALKMD.NET"
FILEO LINKO = "{SCENARIO_DIR}\output\TLOD3.DBF",
    SKIP0=Y,NTLEGS=F
FILEO LINKO[2] = "{SCENARIO_DIR}\output\TLINK_AM_WALK.DBF",
    NTLEGS=N,ONOFFS=T;SKIP0=Y,NTLEGS=F
FILEI ROUTEI[1] = "{SCENARIO_DIR}\output\WALKLBMD.RTE"
FILEI MATI[1] = "{SCENARIO_DIR}\output\TRANSIT.MAT"
FILEO REPORTO = "{SCENARIO_DIR}\output\ASPTR00F.PRN"
FILEO NETO = "{SCENARIO_DIR}\output\TLOADMD1.NET"
PARAMETERS USERCLASSES=1,2,
           TRIPSIJ[1]=(MI.1.OPWALKLOCAL),
           TRIPSIJ[2]=(MI.1.OPWALKPREM)
           NOROUTEERRS=9999999,
           NOROUTEMSGS=0, HDWAYPERIOD=2
REPORT LINES=T ; LINEVOLS=T
PAGEHEIGHT=32767
ENDRUN
```



ASPTR00F.S

```
; Do not change filenames or add or remove FILEI/FILEO statements using an editor. Use
Cube/Application Manager.
```

```
RUN PGM=PUBLIC TRANSPORT PRNFILE="{SCENARIO_DIR}\output\TRANSIT_MDAUTO.PRN" MSG='Off-Peak Period
Auto to Transit Assignment'
FILEI ROUTEI[1] = "{SCENARIO_DIR}\output\AUTOALLMD.RTE"
FILEO LINKO = "{SCENARIO_DIR}\output\TLOD4.DBF",
    SKIP0=Y,NTLEGS=F
FILEO LINKO[2] = "{SCENARIO_DIR}\output\TLINK_AM_AUTO.DBF",
    NTLEGS=N,ONOFFS=T;SKIP0=Y,NTLEGS=F
FILEI NETI = "{SCENARIO_DIR}\output\TNETAUTOMD.NET"
FILEI MATI[1] = "{SCENARIO_DIR}\output\TRANSIT.MAT"
FILEO REPORTO = "{SCENARIO_DIR}\output\ASPTR00H.PRN"
FILEO NETO = "{SCENARIO_DIR}\output\TLOADMD2.NET"
PARAMETERS USERCLASSES=1,
           TRIPSIJ=(MI.1.OPAUTOBA),
           NOROUTEERRS=999999,
           NOROUTEMSGS=0, HDWAYPERIOD=2
REPORT LINES=T ;LINEVOLS=T
PAGEHEIGHT=32767
ENDRUN
```



ASMAT00A.S

```
; Do not change filenames or add or remove FILEI/FILEO statements using an editor. Use
Cube/Application Manager.
RUN PGM=MATRIX
FILEI RECI = "{SCENARIO_DIR}\output\TLOD1.dbf"
FILEO RECO[1] = "{SCENARIO_DIR}\output\LL1.DBF",
  FIELDS=A,B,MODE,NAME,DIST,TIME,SEQ,CNT,HEADWAY_1,VOL
vtot=vtot+ri.VOL
if (ri.SEQ==ri.CNT)
  A       = ri.A
  B       = ri.B
  MODE    = ri.MODE
  NAME    = 'COMBINED'
  DIST    = ri.DIST
  TIME    = ri.TIME
  SEQ     = 1
  CNT     = 1
  HEADWAY= ri.HEADWAY_1
  VOL     = vtot
  vtot    = 0.0
  WRITE RECO=1
endif
ENDRUN
```

ASMAT00B.S

```
; Do not change filenames or add or remove FILEI/FILEO statements using an editor. Use
Cube/Application Manager.
RUN PGM=MATRIX
FILEI RECI = "{SCENARIO_DIR}\output\TLOD2.DBF"
```

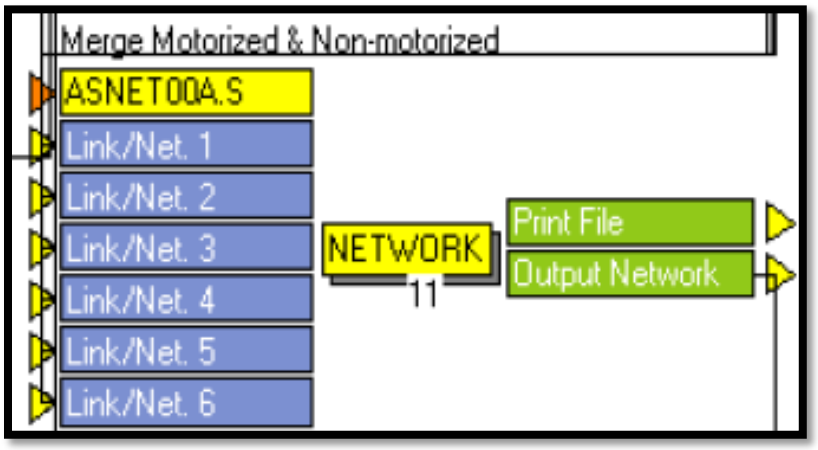
```
FILEO RECO[1] = "{SCENARIO_DIR}\output\LL2.DBF",
  FIELDS=A,B,MODE,NAME,DIST,TIME,SEQ,CNT,HEADWAY_1,VOL
vtot=vtot+ri.VOL
if (ri.SEQ==ri.CNT)
  RO.A       = ri.A
  RO.B       = ri.B
  RO.MODE    = ri.MODE
  RO.NAME    = 'COMBINED'
  RO.DIST    = ri.DIST
  RO.TIME    = ri.TIME
  RO.SEQ     = 1
  RO.CNT     = 1
  RO.HEADWAY= ri.HEADWAY_1
  RO.VOL     = vtot
  vtot    = 0.0
  WRITE RECO=1
endif
ENDRUN
```

ASMAT00C.S

```
; Do not change filenames or add or remove FILEI/FILEO statements using an editor. Use
Cube/Application Manager.
RUN PGM=MATRIX
FILEI RECI = "{SCENARIO_DIR}\output\TLOD3.DBF"
FILEO RECO[1] = "{SCENARIO_DIR}\output\LL3.DBF",
  FIELDS=A,B,MODE,NAME,DIST,TIME,SEQ,CNT,HEADWAY_2,VOL
vtot=vtot+ri.VOL
if (ri.SEQ==ri.CNT)
  RO.A       = ri.A
  RO.B       = ri.B
  RO.MODE    = ri.MODE
  RO.NAME    = 'COMBINED'
  RO.DIST    = ri.DIST
  RO.TIME    = ri.TIME
  RO.SEQ     = 1
  RO.CNT     = 1
  RO.HEADWAY= "ri.HEADWAY_2"
  RO.VOL     = vtot
  vtot    = 0.0
  WRITE RECO=1
endif
ENDRUN
```

ASMAT00D.S

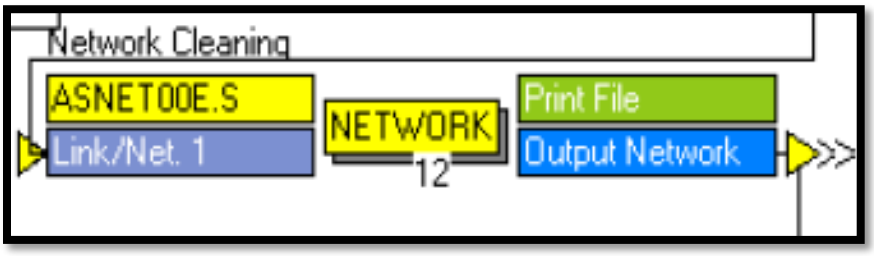
```
; Do not change filenames or add or remove FILEI/FILEO statements using an editor. Use
Cube/Application Manager.
RUN PGM=MATRIX
FILEI RECI = "{SCENARIO_DIR}\output\TLOD4.DBF"
FILEO RECO[1] = "{SCENARIO_DIR}\output\LL4.DBF",
  FIELDS=A,B,MODE,NAME,DIST,TIME,SEQ,CNT,HEADWAY_2,VOL
vtot=vtot+ri.VOL
if (ri.SEQ==ri.CNT)
  RO.A       = ri.A
  RO.B       = ri.B
  RO.MODE    = ri.MODE
  RO.NAME    = 'COMBINED'
  RO.DIST    = ri.DIST
  RO.TIME    = ri.TIME
  RO.SEQ     = 1
  RO.CNT     = 1
  RO.HEADWAY= "ri.HEADWAY_2"
  RO.VOL     = vtot
  vtot    = 0.0
  WRITE RECO=1
endif
ENDRUN
```



ASNET00A.S

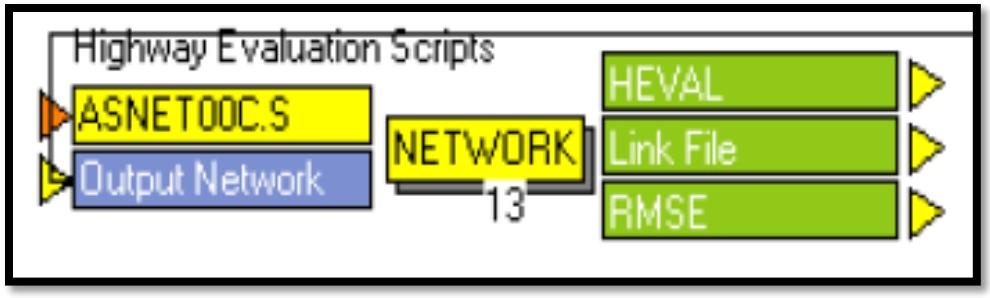
```
; Do not change filenames or add or remove FILEI/FILEO statements using an editor. Use
Cube/Application Manager.
RUN PGM=NETWORK PRNFILE="{SCENARIO_DIR}\output\ASNET00A.PRN" MSG='Merge Motorized & Non-motorized'
FILEO NETO = "{SCENARIO_DIR}\output\COMB_TEMP.NET"
;FILEO PRINTO[1] = "{SCENARIO_DIR}\output\UFOUCH.PRN"
FILEI LINKI[6] = "{SCENARIO_DIR}\output\LL4.DBF"
FILEI LINKI[5] = "{SCENARIO_DIR}\output\LL3.DBF"
FILEI LINKI[4] = "{SCENARIO_DIR}\output\LL2.DBF"
FILEI LINKI[3] = "{SCENARIO_DIR}\output\LL1.DBF"
FILEI LINKI[2] = "{SCENARIO_DIR}\output\NONMOTOR.NET"
FILEI LINKI[1] = "{SCENARIO_DIR}\output\HASSIGN.NET"

PROCESS PHASE=LINKMERGE

  NONMOTORVOL=LI.2.V_1
  CGSPEED=LI.1.CSPD_1
  CGTIME=LI.1.TIME_1
  _UF_LIGHT=LI.1.V1_1
  _UF_HEAVY=LI.1.V2_1;/{PCE_HT} no need to divide by PCE factor now since trucks are treated as
vehicles, not PCE. they are only PCE in the Adjust phase.
  _NON_LIGHT=LI.1.V3_1
  _NON_HEAVY=LI.1.V4_1;/{PCE_HT} no need to divide by PCE factor now since trucks are treated as
vehicles, not PCE. they are only PCE in the Adjust phase.
  _SELZONE_LIGHT=LI.1.V5_1
  _SELZONE_HEAVY=LI.1.V6_1;/{PCE_HT} no need to divide by PCE factor now since trucks are treated
as vehicles, not PCE. they are only PCE in the Adjust phase.
```

```
  SELZONE_MOTOR=_SELZONE_LIGHT+_SELZONE_HEAVY

  UF_MOTOR=_UF_LIGHT+_UF_HEAVY
  LIGHTVEHICLES=_UF_LIGHT+_NON_LIGHT
  HEAVYTRUCKS=_UF_HEAVY+_NON_HEAVY
  MOTORIZEDVOL=LIGHTVEHICLES+HEAVYTRUCKS
  if (MOTORIZEDVOL>0.0)
     UFPCT=100*UF_MOTOR/MOTORIZEDVOL
  endif
  VMT=MOTORIZEDVOL*DISTANCE
  VHT=MOTORIZEDVOL*CGTIME/60.
  PEDESTRIANS=LI.2.V1_1
  BICYCLISTS=LI.2.V2_1
  VOL_CAP=MOTORIZEDVOL/DAILYCAP
  MOTORIZEDVOL2=LI.1.V1T_1+ LI.1.V2T_1/{PCE_HT}+ LI.1.V3T_1+ LI.1.V4T_1/{PCE_HT}

  IF (CAPACITY=0)
     DAILYCAPE=999999
  ELSE
      DAILYCAPE= 10.0*CAPACITY
  ENDIF

  VOL_CAPE=MOTORIZEDVOL/DAILYCAPE

  TranVol=li.3.vol+li.4.vol+li.5.vol+li.6.vol

if(COUNT10 > 0)
  VC=MOTORIZEDVOL/COUNT10
endif

ENDPROCESS
ENDRUN
```



ASNET00E.S

```
; Do not change filenames or add or remove FILEI/FILEO statements using an editor. Use
Cube/Application Manager.
RUN PGM=NETWORK PRNFILE="C:\Gainesville\APPLICATIONS\ASNET00C.PRN" MSG='Network Cleaning'
FILEO NETO = "{SCENARIO_DIR}\output\COMBINEDLOADED.NET",
 EXCLUDE=VOL,MODE,DIST,TIME,SEQ,CNT,HEADWAY_1,HEADWAY_2,                    SECNUM,TWOWAY,
FTYPE1,DIR,ATYPE1,BPRCOEFFICIENT,V_1,VC_1, CSPD_1,VDT_1,VHT_1,V1_1,V2_1,VT_1,V1T_1,
V2T_1,V3_1,V4_1,V5_1,       V6_1,V3T_1,V4T_1,V5T_1,V6T_1,FTYPE1,ATYPE1,       TIME_1,UROADFACTOR,
BPRCOEFFICIENT,BPREXPONENT
FILEI LINKI[1] = "{SCENARIO_DIR}\output\COMB_TEMP.NET"
PROCESS  PHASE=LINKMERGE
; Use this phase to make computations and selections of any data on the LINKI files.
IF( (li.1.count15>0)& l.1.count15<li.1.count10)
count15=count10
ENDIF
ENDPROCESS
ENDRUN
```



ASNET00C.S

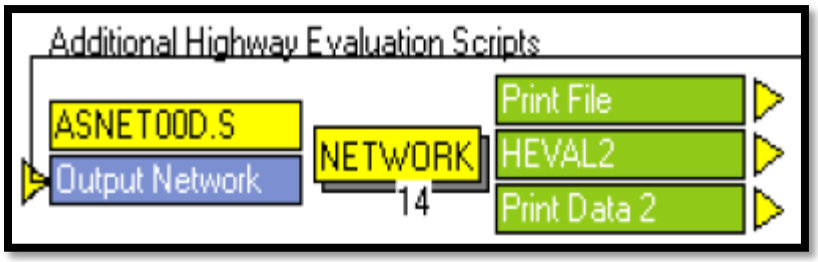
```
; Do not change filenames or add or remove FILEI/FILEO statements using an editor. Use
Cube/Application Manager.
RUN PGM=NETWORK PRNFILE="{SCENARIO_DIR}\output\HEVAl_Daily.PRN" MSG='Highway Evaluation Scripts'
FILEO LINKO = "{SCENARIO_DIR}\output\Daily_Links.DBF"
FILEI LINKI[1] = "{SCENARIO_DIR}\output\COMBINEDLOADED.NET"
FILEO PRINTO[1] = "{SCENARIO_DIR}\output\RMSE.PRN"
; ============================================================
; LINKMERGE PHASE
; ============================================================
PHASE=LINKMERGE
; DUMMY VARIABLES FOR HEVALDBF
ZONE         =1
USECODE      =1
LOCATION     =1
LANDUSE      =1
CCODE        =1
  TOTCNT=li.1.COUNT15
;if (count15<count10) TOTCNT      =li.1.COUNT10

CAP          =LI.1.CAPACITY*LI.1.LANES/LI.1.CONFAC
sl = LI.1.SCRN ; SCREENLINES

   ft=int(li.1.FTYPE/10)
   at=int(li.1.ATYPE/10)
   LNS=li.1.LANES
   TOTAL_VOL=ROUND(li.1.MOTORIZEDVOL)

; initialize arrays and variables
   ARRAY _err=13, _cns=13, _cnt=13, _RGP=13, _volbyft=100, _cntbyft=100
   ARRAY _volbyat=100, _cntbyat=100, _lnkbyft=100, _lnkbyat=100
   ARRAY _capbyft=100, _capbyat=100
   ARRAY _volbysl=100, _cntbysl=100, _lnkbysl=100, _tlnkbysl=100
   _group=(0.0*FT)
   _RGP[1]=1,   _RGP[2]=5000,   _RGP[3]=10000,   _RGP[4]=20000,   _RGP[5]=30000,   _RGP[6]=40000,
_RGP[7]=50000,
   _RGP[8]=60000, _RGP[9]=70000, _RGP[10]=80000, _RGP[11]=90000, _RGP[12]=100000, _RGP[13]=500000
   IF (A=1)
     LOOP _iter=1,13
          _err[_iter]=0,_cnt[_iter]=0,_cns[_iter]=0
     ENDLOOP
     LOOP _iter=1,99
          _volbyft[_iter]=0, _cntbyft[_iter]=0, _lnkbyft[_iter]=0, _capbyft[_iter]=0
          _volbyat[_iter]=0, _cntbyat[_iter]=0, _lnkbyat[_iter]=0, _capbyat[_iter]=0
          _volbysl[_iter]=0, _cntbysl[_iter]=0, _lnkbysl[_iter]=0
     ENDLOOP
   ENDIF

   links=1
   lanemiles=lns*li.1.distance
; calculate and compartmentalize
 if (ft<>8)
   IF(TOTCNT>0)    VOLCNT=TOTAL_VOL/TOTCNT,    NETDIFF=TOTAL_VOL-TOTCNT,    ABSDIFF=ABS(NETDIFF),
ERRORSQ=NETDIFF^2, PCTDIFF=100*NETDIFF/TOTCNT _group=1
   IF (TOTCNT>5000) _group=2
```

```
   IF (TOTCNT>10000) _group=3
   IF (TOTCNT>20000) _group=4
   IF (TOTCNT>30000) _group=5
   IF (TOTCNT>40000) _group=6
   IF (TOTCNT>50000) _group=7
   IF (TOTCNT>60000) _group=8
   IF (TOTCNT>70000) _group=9
   IF (TOTCNT>80000) _group=10
   IF (TOTCNT>90000) _group=11
   IF (TOTCNT>100000) _group=12
   if(SL>0)
     _tlnkbysl[sl]=_tlnkbysl[sl]+1
     _tlnkbysl[100]=_tlnkbysl[100]+1
   endif
   IF (TOTCNT>0)
   _ERR[_group]=ERRORSQ+_ERR[_group],                              _CNS[_group]=TOTCNT+_CNS[_group],
_CNT[_group]=_CNT[_group]+1
   _ERR[13]=ERRORSQ+_ERR[13], _CNS[13]=TOTCNT+_CNS[13], _CNT[13]=_CNT[13]+1
   ENDIF
  endif
IF (TOTCNT>0)
   _volbyft[ft]=_volbyft[ft]+TOTAL_VOL
   _cntbyft[ft]=_cntbyft[ft]+TOTCNT
   _lnkbyft[ft]=_lnkbyft[ft]+1
   _capbyft[ft]=_capbyft[ft]+CAP
   _volbyat[at]=_volbyat[at]+TOTAL_VOL
   _cntbyat[at]=_cntbyat[at]+TOTCNT
   _lnkbyat[at]=_lnkbyat[at]+1
   _capbyat[at]=_capbyat[at]+CAP
   if(SL>0)
     _volbysl[sl]=_volbysl[sl]+TOTAL_VOL
     _cntbysl[sl]=_cntbysl[sl]+TOTCNT
     _lnkbysl[sl]=_lnkbysl[sl]+1
     _volbysl[100]=_volbysl[100]+TOTAL_VOL
     _cntbysl[100]=_cntbysl[100]+TOTCNT
     _lnkbysl[100]=_lnkbysl[100]+1
   endif
   _volbyft[100]=_volbyft[100]+TOTAL_VOL
   _cntbyft[100]=_cntbyft[100]+TOTCNT
   _lnkbyft[100]=_lnkbyft[100]+1
   _volbyat[100]=_volbyat[100]+TOTAL_VOL
   _cntbyat[100]=_cntbyat[100]+TOTCNT
   _lnkbyat[100]=_lnkbyat[100]+1
endif

CROSSTAB  VAR= LINKS LANEMILES, form=14.0c,
    col=FT, range=1-9-1,1-9,
    row=LANES, range=1-9-1,1-9
IF (TOTCNT>0)
    _C_VMT=DISTANCE*TOTCNT
    _A_VMT=DISTANCE*TOTAL_VOL
    _C_VHT=TOTCNT*CGTIME/60.
    _A_VHT=TOTAL_VOL*CGTIME/60.
    _A_VOL=TOTAL_VOL
    _C_VOL=TOTCNT
    _C_CAP=CAP
CROSSTAB  VAR= _A_VOL,_C_VOL,_C_VMT,_A_VMT,_C_VHT,_A_VHT,_C_CAP, form=14.0c,
    col=FT, range=1-4-1,6-9-1,1-9,
    row=AT, range=1-5-1,1-9,
    comp=_A_VOL/_C_VOL,form=8.3,
    comp=_A_VMT/_C_VMT,form=8.3,
    comp=_A_VHT/_C_VHT,form=8.3,
    comp=_A_VOL/_C_CAP,form=8.3
CROSSTAB  VAR= _A_VOL,_C_VOL, form=14.0c,
    col=FT, range=1-4-1,6-9-1,1-9,
    row=LANES, range=1-9-1,1-6

;CROSSTAB  VAR= _A_VOL,_C_VOL,_C_VMT,_A_VMT,_C_VHT,_A_VHT,_C_CAP, form=14.0c,
;    col=FT, range=1-4-1,6-9-1,1-9,
;    row=LOCATION, range=1-7-1,1-7,
;    comp=_A_VOL/_C_VOL,form=8.3,
```

```
;    comp=_A_VMT/_C_VMT,form=8.3,
;    comp=_A_VHT/_C_VHT,form=8.3,
;    comp=_A_VOL/_C_CAP,form=8.3
ENDIF
IF (TOTCNT>0 & SCRN<>0)
    _sVOL=TOTAL_VOL
    _sCNT=TOTCNT
CROSSTAB  VAR= _sVOL,_sCNT, form=9.0c,
    col=FT, range=1-9,
    row=SCRN, range=1-20-1,1-20,
    comp=_sVOL/_sCNT,form=8.3
ENDIF

    _A_VMT_ALL=DISTANCE*TOTAL_VOL
    _A_VHT_ALL=TOTAL_VOL*CGTIME/60.
CROSSTAB  VAR= _A_VMT_ALL, _A_VHT_ALL, form=16.0c,
    col=FT, range=1-4-1,6-9-1,1-9,
    row=AT, range=1-5-1,1-9

;***********PERCENT ERROR BY VOLUME GROUPS CALCULATION************

ARRAY _PCT_ERR_VOLGRP=7
ARRAY _LINKS_VOLGRP=7

IF (_C_VOL<>0)
  IF (_A_VOL >= 1 & _A_VOL< 5000)
  _PCT_ERR_VOLGRP[1]= _PCT_ERR_VOLGRP[1] + ((_A_VOL-_C_VOL)/_C_VOL)*100
  _LINKS_VOLGRP[1]= _LINKS_VOLGRP[1] + 1
  ENDIF

  IF (_A_VOL >= 5000 & _A_VOL< 10000)
  _PCT_ERR_VOLGRP[2]= _PCT_ERR_VOLGRP[2] +  ((_A_VOL-_C_VOL)/_C_VOL)*100
  _LINKS_VOLGRP[2]= _LINKS_VOLGRP[2] + 1
  ENDIF

  IF (_A_VOL >= 10000 & _A_VOL< 20000)
  _PCT_ERR_VOLGRP[3]= _PCT_ERR_VOLGRP[3] +  ((_A_VOL-_C_VOL)/_C_VOL)*100
  _LINKS_VOLGRP[3]= _LINKS_VOLGRP[3] + 1
  ENDIF

  IF (_A_VOL >= 20000 & _A_VOL< 30000)
  _PCT_ERR_VOLGRP[4]= _PCT_ERR_VOLGRP[4] +  ((_A_VOL-_C_VOL)/_C_VOL)*100
  _LINKS_VOLGRP[4]= _LINKS_VOLGRP[4] + 1
  ENDIF

  IF (_A_VOL >= 30000 & _A_VOL< 40000)
  _PCT_ERR_VOLGRP[5]= _PCT_ERR_VOLGRP[5] +  ((_A_VOL-_C_VOL)/_C_VOL)*100
  _LINKS_VOLGRP[5]= _LINKS_VOLGRP[5] + 1
  ENDIF

  IF (_A_VOL >= 40000 & _A_VOL< 50000)
  _PCT_ERR_VOLGRP[6]= _PCT_ERR_VOLGRP[6] +  ((_A_VOL-_C_VOL)/_C_VOL)*100
  _LINKS_VOLGRP[6]= _LINKS_VOLGRP[6] + 1
  ENDIF

  IF (_A_VOL >= 1 & _A_VOL< 500000)
  _PCT_ERR_VOLGRP[7]= _PCT_ERR_VOLGRP[7] +  ((_A_VOL-_C_VOL)/_C_VOL)*100
  _LINKS_VOLGRP[7]= _LINKS_VOLGRP[7] + 1
  ENDIF

ENDIF
ENDPHASE

; ============================================================
; SUMMARY REPORTING
; ============================================================
PHASE=SUMMARY
PRINT LIST="Scenario = {SCENARIO_SHORTNAME}\n" PRINTO=1
; a little loop to write out the Percent Root Mean Square Error
print   list="****************  ALL   COUNT   ROOT   MEAN   SQUARE   ERROR   SUMMARY   (exclude   HOV)
****************", printo=1
```

```
print list=" Group   Volume Range    % RMSE   Target %   Obs", PRINTO=1

LOOP _iter=1,12
  if (_iter=1) _limit='45 - 55'
  if (_iter=2) _limit='35 - 45'
  if (_iter=3) _limit='27 - 35'
  if (_iter=4) _limit='24 - 27'
  if (_iter=5) _limit='22 - 24'
  if (_iter=6) _limit='20 - 22'
  if (_iter=7) _limit='18 - 20'
  if (_iter=8) _limit='17 - 18'
  if (_iter=9) _limit='16 - 17'
  if (_iter=10) _limit='15 - 16'
  if (_iter=11) _limit='14 - 15'
  if (_iter=12) _limit='LT 14  '

 if (_cnt[_iter]>0)
 _RMSE=sqrt(_err[_iter]/(_cnt[_iter]-1))/(_cns[_iter]/_cnt[_iter])*100
 print,
 list=_iter(6.0c)," ",_RGP[_iter](7.0c),"-",_RGP[_iter+1](7.0c)," ",_RMSE(7.1),"%", "  ", _limit,"
",_cnt[_iter](5.0), PRINTO=1
 endif
ENDLOOP
 _iter=13
 _RMSE=sqrt(_err[_iter]/(_cnt[_iter]-1))/(_cns[_iter]/_cnt[_iter])*100
 _limit='32 - 39'
 list=_iter(6.0c),"  ",_RGP[1](7.0c),"-",_RGP[_iter](7.0c),"  ",_RMSE(7.1),"%",  "    ",  _limit,"
",_cnt[_iter](5.0), PRINTO=1

; one for Vol/Cnt by FT
_iter=0
LOOP _iter=1,100
  if (_iter=1) print list="\n","\n ********************** VOLUME AND COUNT SUMMARY BY FACILITY TYPE
**********************", PRINTO=1
if (_cntbyft[_iter]>0) print,
 list="Facility Type Summary for FT=",_iter(3.0c),
 " VOL=",_volbyft[_iter](11.0c),
 " CNT=",_cntbyft[_iter](11.0c),
 " VOL/CNT=",(_volbyft[_iter]/_cntbyft[_iter])(5.2c),
 "   N=",_lnkbyft[_iter](5.0c), PRINTO=1
ENDLOOP

; one for Vol/Cnt by AT
_iter=0
LOOP _iter=1,100
  if (_iter=1) print list="\n","\n ************************ VOLUME AND COUNT SUMMARY BY AREA TYPE
************************", PRINTO=1
if (_cntbyat[_iter]>0) print,
 list="    Area Type Summary for AT=",_iter(3.0c),
 " VOL=",_volbyat[_iter](11.0c),
 " CNT=",_cntbyat[_iter](11.0c),
 " VOL/CNT=",(_volbyat[_iter]/_cntbyat[_iter])(5.2c),
 "   N=",_lnkbyat[_iter](5.0c), PRINTO=1
ENDLOOP

_iter=0
LOOP _iter= 1,7
  if (_iter=1) print list="\n","\n ************************ PERCENT ERROR SUMMARY BY VOLUME GROUP
************************", PRINTO=1
PERCENT_ERROR= _PCT_ERR_VOLGRP[_iter]/_LINKS_VOLGRP[_iter]
PRINT,
 LIST="   VOLUME GROUP=", _iter(3.0c), "   ",
 "   GROUP TOTAL PERCENT ERROR=", _PCT_ERR_VOLGRP[_iter](13.0c),
 "   PERCENT_ERROR=", PERCENT_ERROR(6.2c),
 "   N=", _LINKS_VOLGRP[_iter](11.0c), PRINTO=1
ENDLOOP

;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
;+++ one for Vol/Cnt by SL
_iter=0
LOOP _iter=1,100
```

```
  if (_iter=1) print list="\n","\n *************** VOLUME AND COUNT SUMMARY BY SCREENLINE & CUTLINE
***************", PRINTO=1
    ;read FILE = "C:\TCRPM5\CUBE\SCNAMES.INC"
   SLN=_iter
_RptTitle6= '                                                                  count&Tot  Max Guideline'
_RptTitle6b='  Screen/Cut-Line       Volume        Count     Vol/Count   # Links      Deviation'
_RptTitle6a='  ---------------  -----------   -----------  ---------   ---------   ------------'

  if (_iter=1) print list=_RptTitle6,"\n",_RptTitle6b,"\n",_RptTitle6a," ", printo=1

  if ((_cntbySL[_iter]>0)&(_iter<100))

    _kvol=_cntbySL[_iter]*0.001
    IF(_kvol<100)
      _atol=0.01* ((-0.00005*(_kvol)^3 + 0.013*(_kvol)^2-1.1822*(_kvol)+65.465))
    ELSE
      _atol=2.1783*(_kvol)^-0.4784
    ENDIF
      _rat=(_volbySL[_iter]/_cntbySL[_iter])
    _slerr=abs(1.0-_rat)
    IF(_slerr>_atol)
      _mflg=' ***'
    ELSE
      _mflg='    '
    ENDIF

    print,list=" ",SLN(15.0),
     "   ",_volbySL[_iter](11.0c),
     "  ",_cntbySL[_iter](11.0c),
     "      ",_rat(5.2c),
     "  ",_lnkbySL[_iter](5.0c)," ",_tlnkbySL[_iter](5.0c),_atol(8.2),_mflg PRINTO=1
 ENDIF

 if ((_cntbySL[100]>0)&(_iter=100)) print,
    list="         ", "ALL",
     "       ",_volbySL[_iter](11.0c),
     "  ",_cntbySL[_iter](11.0c),
     "      ",_rat(5.2c),
     "  ",_lnkbySL[_iter](5.0c)," ",_tlnkbySL[_iter](5.0c), PRINTO=1

ENDLOOP
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
ENDPHASE
ENDRUN
```



ASNET00D.S

```
; Do not change filenames or add or remove FILEI/FILEO statements using an editor. Use
Cube/Application Manager.
RUN PGM=NETWORK PRNFILE="{SCENARIO_DIR}\output\HASSIGN2.PRN" MSG='Additional Highway Evaluation
Scripts'
FILEO PRINTO[3] = "{SCENARIO_DIR}\Output\MOE_SUMMARY.PRN"
FILEO PRINTO[2] = "{SCENARIO_DIR}\output\OverallSummary.PRN"
FILEO PRINTO[1] = "{SCENARIO_DIR}\output\HEVAL_Daily2.PRN"
FILEI LINKI[1] = "{SCENARIO_DIR}\output\COMBINEDLOADED.NET"

;=====================================================================
;  LINKMERGE BEGINS
;=====================================================================
```

```
PROCESS PHASE=LINKMERGE

;Calculate working link variables for highway analysis calculations
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
_FT2=li.1.FTYPE                          ;2-digit Facility Type
_AT2=li.1.ATYPE                          ;2-digit Area Type
_FT1=int(_FT2/10)                        ;1-digit Facility Type
_AT1=int(_AT2/10)                        ;1-digit Area Type
_LNS=li.1.LANES                          ;Number of Lanes

_SL=li.1.SCRN                            ;Screenline

_CNT=li.1.COUNT15                        ;Traffic Count
if (count15<count10) _CNT      =li.1.COUNT10
_CAP=li.1.CAPACITY*li.1.LANES/li.1.CONFAC  ;Daily Directional Capacity
_VOL=ROUND(li.1.MOTORIZEDVOL)            ;Estimated Volume

_LNKCNTR=1                               ;Link Counter
if (_CNT>0) _LNK_w_CNT=1                 ;Counter for links w/counts

_MLS=li.1.DISTANCE                       ;Directional System Miles
_LMLS=_LNS*_MLS                          ;Lane Miles
_CGT=li.1.CGTIME                         ;Congested Time
_CGS=li.1.CGSPEED                        ;Congested Speed
_SPD=li.1.SPEED                          ;Free Flow Speed
_WCGS=_CGS*_MLS                          ;Weighted Congested Speed
_WSPD=_SPD*_MLS                          ;Weighted Free Flow Speed

_VMT=_VOL*_MLS                           ;Vehicle Miles Traveled
_CVMT=_CNT*_MLS                          ;Vehicle Miles Traveled using counts
_VHT=_VOL*(_CGT/60)                      ;Vehicle Hours Traveled
_CVHT=_CNT*(_CGT/60)                     ;Vehicle Hours Traveled using counts

if (_CNT>0) _VCNT=_VOL/_CNT              ;Volume over Count
_VCAP=li.1.VOL_CAP                       ;Volume over Capacity

_LIND2=(_AT2*10000)+(_FT2*100 )+(_LNS)     ;2-digit index

;Added for MOE Summary Report
;*************************************************************************
_CRD=li.1.CORRIDOR                       ;Corridor Number
_CNTY=li.1.COUNTY                        ;County
_URBA=li.1.URBANAREA                     ;Urban Area
;_DELAY=((_SPD-_CGS)*_MLS)*_VOL          ;Delay per link
_DELAY=(_WSPD-_WCGS)                       ;Change in SPEED
_TRANVOL=li.1.TRANVOL                    ;Transit Volume

;initialize arrays and variables
  ARRAY _err=13, _cns=13, _count=13, _RGP=13
  ARRAY _volby=999999, _cntby=999999, _vcntby=999999, _volall=999999
  ARRAY _lwctot=999999, _lnktot=999999
  ARRAY _lmiles=999999, _dmiles=999999, _wcgspd=999999, _wffspd=999999
  ARRAY _volvmtval=999999, _cntvmtval=999999, _volvhtval=999999, _cntvhtval=999999
  ARRAY _vmtall=999999, _vhtall=999999
  ARRAY _slvol=99, _slcnt=99
  ARRAY _crdvht=999999                   ;added for MOE Summary
  ARRAY _crdad=999999                    ;added for MOE Summary
  ARRAY _cntyad=999999                   ;added for MOE Summary
  ARRAY _urbaad=999999                   ;added for MOE Summary
  ARRAY _vcap8=999999                    ;added for MOE Summary
  ARRAY _crdtotv=999999                  ;added for MOE Summary
  ARRAY _crdtvol=999999                  ;added for MOE Summary
  ARRAY _cntytvol=999999                 ;added for MOE Summary
  ARRAY _cntytotv=999999                 ;added for MOE Summary
  ARRAY _urbatvol=999999                 ;added for MOE Summary
  ARRAY _urbatotv=999999                 ;added for MOE Summary

   _lnktot[_LIND2]=_lnktot[_LIND2]+_LNKCNTR
   _lmiles[_LIND2]=_lmiles[_LIND2]+_LMLS
   _dmiles[_LIND2]=_dmiles[_LIND2]+_MLS
```

```
   _volall[_LIND2]=_volall[_LIND2]+_VOL
   _wcgspd[_LIND2]=_wcgspd[_LIND2]+_WCGS
   _wffspd[_LIND2]=_wffspd[_LIND2]+_WSPD
   _vmtall[_LIND2]=_vmtall[_LIND2]+_VMT
   _vhtall[_LIND2]=_vhtall[_LIND2]+_VHT

IF (_CNT>0)                                        ;Variables only for link with counts
   _volby[_LIND2]=_volby[_LIND2]+_VOL
   _cntby[_LIND2]=_cntby[_LIND2]+_CNT
   _vcntby[_LIND2]=_vcntby[_LIND2]+_VCNT
   _lwctot[_LIND2]=_lwctot[_LIND2]+_LNK_w_CNT
   _volvmtval[_LIND2]=_volvmtval[_LIND2]+_VMT
   _cntvmtval[_LIND2]=_cntvmtval[_LIND2]+_CVMT
   _volvhtval[_LIND2]=_volvhtval[_LIND2]+_VHT
   _cntvhtval[_LIND2]=_cntvhtval[_LIND2]+_CVHT
   _slvol[_SL]=_slvol[_SL]+_VOL
   _slcnt[_SL]=_slcnt[_SL]+_CNT
ENDIF
;*************************************************************************
;Added Variables for MOE Summary Report
;*************************************************************************
 _crdvht[_CRD]=_crdvht[_CRD]+_VHT
  IF (_VMT>0)
 ;  IF (_DELAY>0)
 _crdad[_CRD]=_crdad[_CRD]+_DELAY*_VMT
 _cntyad[_CNTY]=_cntyad[_CNTY]+_DELAY*_VMT
 _urbaad[_URBA]=_urbaad[_URBA]+_DELAY*_VMT
 ;   ELSEIF (_DELAY=0)
 ;_crdad[_CRD]=_crdad[_CRD]+_VMT
 ;_cntyad[_CNTY]=_cntyad[_CNTY]+_VMT
 ;_urbaad[_URBA]=_urbaad[_URBA]+_VMT
 ;ENDIF
 ENDIF
 IF (_VCAP>1.10)
 _vcap8[_VCAP]=_vcap8[_VCAP]+(_TRANVOL/_MLS)
 ENDIF
  IF (_TRANVOL>0)
  _crdtvol[_CRD]=_crdtvol[_CRD]+_TRANVOL
 _crdtotv[_CRD]=_crdtotv[_CRD]+(_VOL*1.38)+_TRANVOL
 _cntytvol[_CNTY]=_cntytvol[_CNTY]+_TRANVOL
 _cntytotv[_CNTY]=_cntytotv[_CNTY]+(_VOL*1.38)+_TRANVOL
 _urbatvol[_URBA]=_urbatvol[_URBA]+_TRANVOL
 _urbatotv[_URBA]=_urbatotv[_URBA]+(_VOL*1.38)+_TRANVOL
 ENDIF
 ;*************************************************************************
ENDPROCESS

;~~~~~~~~~~~~~~~~BEGIN REPORTING PORTION OF SCRIPT~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

PROCESS PHASE=SUMMARY

;*************PLACE HOLDER FOR VALIDATE/ANALYSIS SWITCH************

;---------------------------------------------------------------------
; VALIDATION VERSION OF HIGHWAY ANALYSIS BEGINS HERE
;---------------------------------------------------------------------
print list= "Highway Analysis and Evaluation Report---Alpha Version 0.1", PRINTO=1
print list= '@date.rundate@', printo=1
print list= '@time.runtime@', PRINTO=1
;print list= 'Date: ',@date.rundate@, printo=1
;print list= 'Time: ',@time.runtime@, PRINTO=1
;print list= "\n","\n","\n", PRINTO=1
;print list= " ", PRINTO=1
;print list= " ", PRINTO=1
;print list= "Facility Types",
;       " ",
;       "\n", "10    Generic Freeway",
;       "\n", "11    Urban Freeway Group 1",
;       "\n", "12    Other Freeway not in Group 1",
;       "\n", "15    Collector/Distributor Lanes",
;       "\n", "16    Controlled Access Expressway",
```

```
;       "\n", "17    Controlled Access Parkway",
;       "\n", "20    Generic Divided Arterial",
;       "\n", "21    Divided Arterial Unsignalized (55 mph)",
;       "\n", "22    Divided Arterial Unsignalized (45 mph)",
;       "\n", "23    Divided Arterial Class Ia",
;       "\n", "24    Divided Arterial Class Ib",
;       "\n", "25    Divided Arterial Class II/III",
;       "\n", "30    Generic Undivided Arterial",
;       "\n", "31    Undivided Arterial Unsignalized with Turn Bays",
;       "\n", "32    Undivided Arterial Class Ia with Turn Bays",
;       "\n", "33    Undivided Arterial Class Ib with Turn Bays",
;       "\n", "34    Undivided Arterial Class II/III with Turn Bays",
;       "\n", "35    Undivided Arterial Unsignalized without Turn Bays",
;       "\n", "36    Undivided Arterial Class Ia without Turn Bays",
;       "\n", "37    Undivided Arterial Class Ib without Turn Bays",
;       "\n", "38    Undivided Arterial Class II/III without Turn Bays",
;       "\n", "40    Generic Collector",
;       "\n", "41    Major Local Divided Roadway",
;       "\n", "42    Major Local Undivided Roadway with Turn Bays",
;       "\n", "43    Major Local Undivided Roadway without Turn Bays",
;       "\n", "44    Other Local Divided Roadway",
;       "\n", "45    Other Local Undivided Roadway with Turn Bays",
;       "\n", "46    Other Local Undivided Roadway without Turn Bays",
;       "\n", "47    Low Speed Local Collector",
;       "\n", "48    Very Low Speed Local Collector",
;       "\n", "50    Generic Centroid Connector",
;       "\n", "51    Basic Centroid Connector",
;       "\n", "52    External Station Centroid Connector",
;       "\n", "60    Generic One-Way",
;       "\n", "61    One-Way Facility Unsignalized",
;       "\n", "62    One-Way Facility Class Ia",
;       "\n", "63    One-Way Facility Class Ib",
;       "\n", "64    One-Way Facility Class II/III",
;       "\n", "65    Frontage Road Unsignalized",
;       "\n", "66    Frontage Road Class Ia",
;       "\n", "67    Frontage Road Class Ib",
;       "\n", "68    Frontage Road Class II/III",
;       "\n", "70    Generic Ramp",
;       "\n", "71    Freeway On-Ramp",
;       "\n", "72    Freeway Loop On-Ramp",
;       "\n", "73    Other On-Ramp",
;       "\n", "74    Other Loop On-Ramp",
;       "\n", "75    Freeway Off-Ramp",
;       "\n", "76    Freeway Loop Off-Ramp",
;       "\n", "77    Other Off-Ramp",
;       "\n", "78    Other Loop Off-Ramp",
;       "\n", "79    Freeway-Freeway High-Speed Ramp",
;  PRINTO=1


;=============================================================================================
; BEGIN NUMBER OF LINKS REPORT ------------ X = NUMBER OF LINKS
;=============================================================================================
Print list=" ", PRINTO=1
Print
list="*************************************************************************************
***********************************", PRINTO=1
Print                                                                                 list="*
*", PRINTO=1
Print list="*                                    Number of Directional Links (Centroid Connectors
Excluded)                                    *", PRINTO=1
Print                                                                                 list="*
*", PRINTO=1
Print
list="*************************************************************************************
***********************************", PRINTO=1
Print list=" ", PRINTO=1
;-----------------2-DIGIT FACILITY TYPES BY 2-DIGIT AREA TYPES--------------------

LOOP _a1iter=100000,599999,100000                        ;^Begin Loop 1: Cycles through Area Types
(ATYPE) by 10
```

```
  _aat1=int(_a1iter/100000)                                    ; in order to get single digit ATYPE.
  print list= "Area Type ",_aat1(1.0),"x Range:",
           "\n ", PRINTO=1

  LOOP _aiter=_a1iter,599999,10000                             ;^Begin Loop 2: Cycles through ATYPE by
1
    if (_aiter>_a1iter+99999) BREAK                            ; in order to get two-digit ATYPE.
    _aat2=int(_aiter/10000)

    _avcheck=0                                                 ;^Initialize ATYPE X checking variable.

    LOOP _achkiter=_aiter,599999,1                             ;^Begin Loop 3: Cycles through Lanes and
Facility Types (FTYPE)
      if (_achkiter>_aiter+9999) BREAK                         ; for current ATYPE in Loop 2 and totals
X checking variable.
      _avcheck=_avcheck+_lnktot[_achkiter]
    ENDLOOP                                                    ;^End Loop 3.

    if (_avcheck>0)                                            ;^Begin Condition 1: If current ATYPE in
Loop 2
                                                               ; has X>0 continue to report X. Else skip
ATYPE.
      _supertotal=0                                            ;^Initialize ATYPE total X.

      Print list= "Area Type ",_aat2(2.0), PRINTO=1            ;^Header
      Print list= "                                                    Number of Lanes per Direction
", PRINTO=1
      Print list= "FType          1           2           3           4           5           6
7           8           9       Totals", PRINTO=1
      Print list= "-------------------------------------------------------------------------------
-----------------------------------------------", PRINTO=1

      LOOP _fiter=100,9900,100                                 ;^Begin Loop 4: Cycles through FTYPE
                                                               ; by 1 in order to get two-digit FTYPE.
       _vcheck=0                                               ;^Initialize FTYPE X checking variable.

       LOOP _liter=1,9,1                                       ;^Begin Loop 5: Cycles through Lanes for
current
         _vcheck=_vcheck+_lnktot[_aiter+_fiter+_liter]    ; FTYPE in Loop 4 and totals X checking
variable.
       ENDLOOP                                                 ;^End Loop 5.

        if (_vcheck>0 & (_fiter<5000 | _fiter>5999))           ;^Begin Condition 2: If current FTYPE in
Loop 4
          _fft2=int(_fiter/100)                               ; has X>0 continue to report X. Else skip
FTYPE.
          print list= _fft2(2.0),"    ", PRINTO=1
          _totvols=0                                           ;^Initialize FTYPE total X.

          LOOP _liter2=1,9,1                                   ;^Begin Loop 6: Cycles through Lanes to
generate ATYPE by FTYPE by Lanes total X.
            print list="\\"," ",_lnktot[_aiter+_fiter+_liter2](10.0C)," ", PRINTO=1
            _totvols=_totvols+_lnktot[_aiter+_fiter+_liter2]
            _supertotal=_supertotal+_lnktot[_aiter+_fiter+_liter2]
          ENDLOOP                                              ;^End Loop 6.

            print list="\\"," ",_totvols(10.0C), PRINTO=1
        endif                                                  ;^End Condition 2.

      ENDLOOP                                                  ;^End Loop 4.

      Print list= "-------------------------------------------------------------------------------
-----------------------------------------------", PRINTO=1
      print list="Totals", PRINTO=1

      LOOP _liter3=1,9,1                                       ;^Begin Loop 7: Cycles through Lanes for
                                                               ; current ATYPE in Loop 2.
        _lntotals=0                                            ;^Initialize Lane total X.

        LOOP _aiter2=_aiter,599999,100                         ;^Begin Loop 8: Cycles through FTYPE for
current ATYPE
```

```
          if (_aiter2>_aiter+9999) BREAK                    ; in Loop 2 to generate Lane total X.
          if (_aiter2<_aiter+5000 | _aiter2>_aiter+5999)
            _lntotiter=_aiter2+_liter3
            _lntotals=_lntotals+_lnktot[_lntotiter]
          endif
        ENDLOOP                                             ;^End Loop 8

        print list="\\"," ",_lntotals(10.0C)," ", PRINTO=1
      ENDLOOP                                               ;^End Loop 7

      print list="\\"," ",_supertotal(10.0C), PRINTO=1
      print list=" ", PRINTO=1
    endif                                                   ;^End Condition 1.

  ENDLOOP                                                   ;^End Loop 2.

  print list=" ", PRINTO=1
ENDLOOP                                                     ;^End Loop 1.

;-----------------2-DIGIT FACILITY TYPES BY TOTAL AREA TYPES--------------------

Print list= "Total Area Types ", PRINTO=1                   ;^Header
Print list= "                                                        Number of Lanes per Direction
", PRINTO=1
Print list= "FType         1           2           3           4           5           6           7
8           9       Totals", PRINTO=1
Print list= "-------------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1

LOOP _fiter2=100,9900,100                                  ;^Begin Loop 9: Cycles through FTYPES to
get
  _fft2=int(_fiter2/100)                                    ; two-digit FTYPE.

  _tafvcheck=0                                              ;^Initialize FTYPE X checking variable.
 if (_fft2<50 | _fft2>59)
  LOOP _liter5=1,9,1                                          ;^Begin Loop 10: Cycles through Lanes
for current
                                                            ; FTYPE in Loop 9.
    LOOP _aiter4= 100000,599999,10000                         ;^Begin Loop 11: Cycles through ATYPE
for
      _tafvcheck=_tafvcheck+_lnktot[_aiter4+_fiter2+_liter5] ; current Lanes and FTYPE in order to
total X checking variable.
    ENDLOOP                                                   ;^End Loop 11.

  ENDLOOP                                                   ;^End Loop 10.

  if (_tafvcheck>0)                                        ;^Begin Condition 3: If current FTYPE in
Loop 9
    print list= _fft2(2.0),"    ", PRINTO=1               ; has X>0 continue to report X. Else skip
FTYPE.

    LOOP _liter4= 1,9,1                                       ;^Begin Loop 12: Cycles through Lanes
for current FTYPE
                                                            ; in Loop 9.
      _totftat=0                                           ;^Initialize FTYPE total X for all ATYPE.

      LOOP _aiter3= 100000,599999,10000                       ;^Begin Loop 13: Cycles through ATYPE
for current Lanes in Loop 12
        _totftat=_totftat+_lnktot[_aiter3+_fiter2+_liter4] ; in order to generate total X for FTYPE
by Lane for all ATYPE.
      ENDLOOP                                               ;^End Loop 13.

      print list="\\"," ",_totftat(10.0C)," ", PRINTO=1
    ENDLOOP                                                 ;^End Loop 12.

    print list="\\"," ",_tafvcheck(10.0C), PRINTO=1
  endif                                                     ;^End Condition 3.
 endif
ENDLOOP                                                     ;^End Loop 9.
```

```
Print list= "-------------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1
print list="Totals", PRINTO=1

_supertotal=0                                          ;^Initialize all ATYPE total X.

LOOP _liter6=1,9,1                                     ;^Begin Loop 14: Cycles through Lanes.

  _lntotals=0                                          ;^Initialize total X for Lanes.

  LOOP _aiter5=100000,599999,100                         ;^Begin Loop 15: Cycles through ATYPE
and
    if ((_aiter5<105000 | _aiter5>105999) &
        (_aiter5<115000 | _aiter5>115999) &
        (_aiter5<125000 | _aiter5>125999) &
        (_aiter5<135000 | _aiter5>135999) &
        (_aiter5<145000 | _aiter5>145999) &
        (_aiter5<155000 | _aiter5>155999) &
        (_aiter5<165000 | _aiter5>165999) &
        (_aiter5<175000 | _aiter5>175999) &
        (_aiter5<185000 | _aiter5>185999) &
        (_aiter5<195000 | _aiter5>195999) &
        (_aiter5<205000 | _aiter5>205999) &
        (_aiter5<215000 | _aiter5>215999) &
        (_aiter5<225000 | _aiter5>225999) &
        (_aiter5<235000 | _aiter5>235999) &
        (_aiter5<245000 | _aiter5>245999) &
        (_aiter5<255000 | _aiter5>255999) &
        (_aiter5<265000 | _aiter5>265999) &
        (_aiter5<275000 | _aiter5>275999) &
        (_aiter5<285000 | _aiter5>285999) &
        (_aiter5<295000 | _aiter5>295999) &
        (_aiter5<305000 | _aiter5>305999) &
        (_aiter5<315000 | _aiter5>315999) &
        (_aiter5<325000 | _aiter5>325999) &
        (_aiter5<335000 | _aiter5>335999) &
        (_aiter5<345000 | _aiter5>345999) &
        (_aiter5<355000 | _aiter5>355999) &
        (_aiter5<365000 | _aiter5>365999) &
        (_aiter5<375000 | _aiter5>375999) &
        (_aiter5<385000 | _aiter5>385999) &
        (_aiter5<395000 | _aiter5>395999) &
        (_aiter5<405000 | _aiter5>405999) &
        (_aiter5<415000 | _aiter5>415999) &
        (_aiter5<425000 | _aiter5>425999) &
        (_aiter5<435000 | _aiter5>435999) &
        (_aiter5<445000 | _aiter5>445999) &
        (_aiter5<455000 | _aiter5>455999) &
        (_aiter5<465000 | _aiter5>465999) &
        (_aiter5<475000 | _aiter5>475999) &
        (_aiter5<485000 | _aiter5>485999) &
        (_aiter5<495000 | _aiter5>495999) &
        (_aiter5<505000 | _aiter5>505999) &
        (_aiter5<515000 | _aiter5>515999) &
        (_aiter5<525000 | _aiter5>525999) &
        (_aiter5<535000 | _aiter5>535999) &
        (_aiter5<545000 | _aiter5>545999) &
        (_aiter5<555000 | _aiter5>555999) &
        (_aiter5<565000 | _aiter5>565999) &
        (_aiter5<575000 | _aiter5>575999) &
        (_aiter5<585000 | _aiter5>585999) &
        (_aiter5<595000 | _aiter5>595999))

      _lntotiter=_aiter5+_liter6                       ; FTYPE in order to generate total X for
      _lntotals=_lntotals+_lnktot[_lntotiter]           ; Lanes.

    endif
  ENDLOOP                                              ;^End Loop 15.

  print list="\\"," ",_lntotals(10.0C)," ", PRINTO=1
  _supertotal=_supertotal+_lntotals                    ;^Generate total X for all ATYPE.
```

```
ENDLOOP                                                    ;^End Loop 14.
print list="\\"," ",_supertotal(10.0C), PRINTO=1
print list=" ","\n ", PRINTO=1

;---------------1-DIGIT FACILITY TYPES BY 1-DIGIT AREA TYPES SUMMARY---------------

Print list= "Total Summary Area Types by Facility Types ", PRINTO=1 ;^Header
Print list= "                                                    Single Digit Facility Types
", PRINTO=1
Print list= "AType         1x          2x          3x          4x          5x          6x          7x
8x          9x      Totals", PRINTO=1
Print list= "-------------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1

LOOP _a1iter2=100000,599999,100000                        ;^Begin Loop 16: Cycles through ATYPE by
10 to
  _aat1=int(_a1iter2/100000)                               ; get single digit ATYPE.
  print list= _aat1(1.0),"x","    ", PRINTO=1

  _fttotal=0                                               ;^Initialize total X for all ATYPE

    LOOP _f1iter=1000,9900,1000                            ;^Begin Loop 17: Cycles through FTYPE by
10 to
                                                           ; get single digit FTYPE.
      _totftlns=0                                            ;^Initialize total X for all FTYPE by
all Lanes.
      if (_f1iter<5000 | _f1iter>5999)
      LOOP _fiter3=_f1iter,9900,100                        ;^Begin Loop 18: Cycles through two-digit
FTYPE
        if (_fiter3>_f1iter+999) BREAK                     ; for current single digit FTYPE in Loop
17.

        LOOP _aiter6=_a1iter2,599999,10000                 ;^Begin Loop 19: Cycles through two-digit
ATYPE
          if (_aiter6>_a1iter2+99999) BREAK                ; for current single digit ATYPE in Loop
16.

            LOOP _liter7=1,9,1                                       ;^Begin Loop 20: Cycles through
Lanes for current FTYPE and ATYPE
              _totftlns=_totftlns+_lnktot[_aiter6+_fiter3+_liter7]   ; in order to generate total
X for FTYPE by ATYPE.
            ENDLOOP                                                   ;^End Loop 20.

        ENDLOOP                                            ;^End Loop 19.

      ENDLOOP                                              ;^End Loop 18.
      endif
      _fttotal=_fttotal+_totftlns                          ;^Generate total X for ATYPE.

      print list="\\"," ",_totftlns(10.0C)," ", PRINTO=1
    ENDLOOP                                                ;^End Loop 17.

  print list="\\"," ",_fttotal(10.0c), PRINTO=1
ENDLOOP                                                    ;^End Loop 16.

Print list= "-------------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1
print list="Totals", PRINTO=1

_supertotal=0                                              ;^Initialize overall total X.

LOOP _f1iter2=1000,9900,1000                               ;^Begin Loop 21: Cycles through FTYPE by
10
                                                           ; to get single digit FTYPE.
  _ftotals=0                                               ;^Initialize total X by FTYPE

  LOOP _fiter4=_f1iter2,9900,100                           ;^Begin Loop 22: Cycles through FTYPE by
1 to
    if (_fiter4>_f1iter2+999) BREAK                          ; get all two-digit FTYPE for current
FTYPE in
```

```
    if (_f1iter2<5000 | _f1iter2>5999)                                                    ; Loop
21.
    LOOP _liter8=1,9,1                                    ;^Begin Loop 23: Cycles through Lanes.

      LOOP _aiter7=100000,599999,10000                    ;^Begin Loop 24: Cycles through ATYPE in
order
        _ftotiter=_aiter7+_fiter4+_liter8                   ; to generate total X by single digit
FTYPE.
        _ftotals=_ftotals+_lnktot[_ftotiter]
      ENDLOOP                                             ;^End Loop 24.

    ENDLOOP                                               ;^End Loop 23.
    endif
  ENDLOOP                                                 ;^End Loop 22.
  _supertotal=_supertotal+_ftotals                        ;^Generate overall total for all single
digit ATYPE
                                                          ; by all single digit FTYPE.
  print list="\\"," ",_ftotals(10.0C)," ", PRINTO=1
ENDLOOP                                                   ;^End Loop 21.
_numlinks=_supertotal
print list="\\"," ",_supertotal(10.0C), PRINTO=1
print list=" ", PRINTO=1
;************************************************************************
; END NUMBER OF LINKS REPORT
;************************************************************************

;=============================================================================================
; BEGIN DIRECTIONAL MILES REPORT ------------ X = DIRECTIONAL MILES
;=============================================================================================
Print list=" ", PRINTO=1
Print
list="*************************************************************************************************
***********************************", PRINTO=1
Print                                                                                    list="*
*", PRINTO=1
Print list="*                                    Directional Miles (Centroid Connectors Excluded)
*", PRINTO=1
Print                                                                                    list="*
*", PRINTO=1
Print
list="*************************************************************************************************
***********************************", PRINTO=1
Print list=" ", PRINTO=1
;-----------------2-DIGIT FACILITY TYPES BY 2-DIGIT AREA TYPES--------------------

LOOP _a1iter=100000,599999,100000                         ;^Begin Loop 1: Cycles through Area Types
(ATYPE) by 10
  _aat1=int(_a1iter/100000)                                ; in order to get single digit ATYPE.
  print list= "Area Type ",_aat1(1.0),"x Range:",
           "\n ", PRINTO=1

  LOOP _aiter=_a1iter,599999,10000                         ;^Begin Loop 2: Cycles through ATYPE by
1
    if (_aiter>_a1iter+99999) BREAK                        ; in order to get two-digit ATYPE.
    _aat2=int(_aiter/10000)

    _avcheck=0                                             ;^Initialize ATYPE X checking variable.

    LOOP _achkiter=_aiter,599999,1                         ;^Begin Loop 3: Cycles through Lanes and
Facility Types (FTYPE)
      if (_achkiter>_aiter+9999) BREAK                     ; for current ATYPE in Loop 2 and totals
X checking variable.
      _avcheck=_avcheck+_dmiles[_achkiter]
    ENDLOOP                                                ;^End Loop 3.

    if (_avcheck>0)                                        ;^Begin Condition 1: If current ATYPE in
Loop 2
                                                           ; has X>0 continue to report X. Else skip
ATYPE.
      _supertotal=0                                        ;^Initialize ATYPE total X.
```

```
      Print list= "Area Type ",_aat2(2.0), PRINTO=1       ;^Header
      Print list= "                                                Number of Lanes per Direction
", PRINTO=1
      Print list= "FType          1           2           3           4           5           6
7           8           9       Totals", PRINTO=1
      Print list= "-------------------------------------------------------------------------------
-----------------------------------------------", PRINTO=1

      LOOP _fiter=100,9900,100                            ;^Begin Loop 4: Cycles through FTYPE
                                                          ; by 1 in order to get two-digit FTYPE.
       _vcheck=0                                          ;^Initialize FTYPE X checking variable.

       LOOP _liter=1,9,1                                 ;^Begin Loop 5: Cycles through Lanes for
current
         _vcheck=_vcheck+_dmiles[_aiter+_fiter+_liter]    ; FTYPE in Loop 4 and totals X checking
variable.
       ENDLOOP                                            ;^End Loop 5.

        if (_vcheck>0 & (_fiter<5000 | _fiter>5999))      ;^Begin Condition 2: If current FTYPE in
Loop 4
          _fft2=int(_fiter/100)                          ; has X>0 continue to report X. Else skip
FTYPE.
          print list= _fft2(2.0),"    ", PRINTO=1
          _totvols=0                                      ;^Initialize FTYPE total X.

          LOOP _liter2=1,9,1                               ;^Begin Loop 6: Cycles through Lanes to
generate ATYPE by FTYPE by Lanes total X.
            print list="\\"," ",_dmiles[_aiter+_fiter+_liter2](10.2C)," ", PRINTO=1
            _totvols=_totvols+_dmiles[_aiter+_fiter+_liter2]
            _supertotal=_supertotal+_dmiles[_aiter+_fiter+_liter2]
          ENDLOOP                                         ;^End Loop 6.

            print list="\\"," ",_totvols(10.2C), PRINTO=1
        endif                                             ;^End Condition 2.

      ENDLOOP                                             ;^End Loop 4.

      Print list= "-------------------------------------------------------------------------------
-----------------------------------------------", PRINTO=1
      print list="Totals", PRINTO=1

      LOOP _liter3=1,9,1                                 ;^Begin Loop 7: Cycles through Lanes for
                                                         ; current ATYPE in Loop 2.
        _lntotals=0                                      ;^Initialize Lane total X.

        LOOP _aiter2=_aiter,599999,100                   ;^Begin Loop 8: Cycles through FTYPE for
current ATYPE
          if (_aiter2>_aiter+9999) BREAK                  ; in Loop 2 to generate Lane total X.
          if (_aiter2<_aiter+5000 | _aiter2>_aiter+5999)
            _lntotiter=_aiter2+_liter3
            _lntotals=_lntotals+_dmiles[_lntotiter]
          endif
        ENDLOOP                                          ;^End Loop 8

        print list="\\"," ",_lntotals(10.2C)," ", PRINTO=1
      ENDLOOP                                            ;^End Loop 7

      print list="\\"," ",_supertotal(10.2C), PRINTO=1
      print list=" ", PRINTO=1
    endif                                                ;^End Condition 1.

  ENDLOOP                                                ;^End Loop 2.

  print list=" ", PRINTO=1
ENDLOOP                                                  ;^End Loop 1.

;-----------------2-DIGIT FACILITY TYPES BY TOTAL AREA TYPES--------------------

Print list= "Total Area Types ", PRINTO=1                ;^Header
Print list= "                                                Number of Lanes per Direction
", PRINTO=1
```

```
Print list= "FType         1           2           3           4           5           6           7
8           9       Totals", PRINTO=1
Print list= "-------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1

LOOP _fiter2=100,9900,100                                    ;^Begin Loop 9: Cycles through FTYPES to
get
  _fft2=int(_fiter2/100)                                     ; two-digit FTYPE.

  _tafvcheck=0                                               ;^Initialize FTYPE X checking variable.
 if (_fft2<50 | _fft2>59)
  LOOP _liter5=1,9,1                                           ;^Begin Loop 10: Cycles through Lanes
for current
                                                             ; FTYPE in Loop 9.
    LOOP _aiter4= 100000,599999,10000                          ;^Begin Loop 11: Cycles through ATYPE
for
      _tafvcheck=_tafvcheck+_dmiles[_aiter4+_fiter2+_liter5] ; current Lanes and FTYPE in order to
total X checking variable.
    ENDLOOP                                                    ;^End Loop 11.

  ENDLOOP                                                    ;^End Loop 10.

  if (_tafvcheck>0)                                          ;^Begin Condition 3: If current FTYPE in
Loop 9
    print list= _fft2(2.0),"    ", PRINTO=1                  ; has X>0 continue to report X. Else skip
FTYPE.

    LOOP _liter4= 1,9,1                                        ;^Begin Loop 12: Cycles through Lanes
for current FTYPE
                                                             ; in Loop 9.
        _totftat=0                                           ;^Initialize FTYPE total X for all ATYPE.

      LOOP _aiter3= 100000,599999,10000                        ;^Begin Loop 13: Cycles through ATYPE
for current Lanes in Loop 12
        _totftat=_totftat+_dmiles[_aiter3+_fiter2+_liter4] ; in order to generate total X for FTYPE
by Lane for all ATYPE.
      ENDLOOP                                                ;^End Loop 13.

      print list="\\"," ",_totftat(10.2C)," ", PRINTO=1
    ENDLOOP                                                  ;^End Loop 12.

    print list="\\"," ",_tafvcheck(10.2C), PRINTO=1
  endif                                                      ;^End Condition 3.
 endif
ENDLOOP                                                      ;^End Loop 9.

Print list= "-------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1
print list="Totals", PRINTO=1

_supertotal=0                                                ;^Initialize all ATYPE total X.

LOOP _liter6=1,9,1                                           ;^Begin Loop 14: Cycles through Lanes.

  _lntotals=0                                                ;^Initialize total X for Lanes.

  LOOP _aiter5=100000,599999,100                               ;^Begin Loop 15: Cycles through ATYPE
and
    if ((_aiter5<105000 | _aiter5>105999) &
        (_aiter5<115000 | _aiter5>115999) &
        (_aiter5<125000 | _aiter5>125999) &
        (_aiter5<135000 | _aiter5>135999) &
        (_aiter5<145000 | _aiter5>145999) &
        (_aiter5<155000 | _aiter5>155999) &
        (_aiter5<165000 | _aiter5>165999) &
        (_aiter5<175000 | _aiter5>175999) &
        (_aiter5<185000 | _aiter5>185999) &
        (_aiter5<195000 | _aiter5>195999) &
        (_aiter5<205000 | _aiter5>205999) &
        (_aiter5<215000 | _aiter5>215999) &
        (_aiter5<225000 | _aiter5>225999) &
```

```
          (_aiter5<235000 | _aiter5>235999) &
          (_aiter5<245000 | _aiter5>245999) &
          (_aiter5<255000 | _aiter5>255999) &
          (_aiter5<265000 | _aiter5>265999) &
          (_aiter5<275000 | _aiter5>275999) &
          (_aiter5<285000 | _aiter5>285999) &
          (_aiter5<295000 | _aiter5>295999) &
          (_aiter5<305000 | _aiter5>305999) &
          (_aiter5<315000 | _aiter5>315999) &
          (_aiter5<325000 | _aiter5>325999) &
          (_aiter5<335000 | _aiter5>335999) &
          (_aiter5<345000 | _aiter5>345999) &
          (_aiter5<355000 | _aiter5>355999) &
          (_aiter5<365000 | _aiter5>365999) &
          (_aiter5<375000 | _aiter5>375999) &
          (_aiter5<385000 | _aiter5>385999) &
          (_aiter5<395000 | _aiter5>395999) &
          (_aiter5<405000 | _aiter5>405999) &
          (_aiter5<415000 | _aiter5>415999) &
          (_aiter5<425000 | _aiter5>425999) &
          (_aiter5<435000 | _aiter5>435999) &
          (_aiter5<445000 | _aiter5>445999) &
          (_aiter5<455000 | _aiter5>455999) &
          (_aiter5<465000 | _aiter5>465999) &
          (_aiter5<475000 | _aiter5>475999) &
          (_aiter5<485000 | _aiter5>485999) &
          (_aiter5<495000 | _aiter5>495999) &
          (_aiter5<505000 | _aiter5>505999) &
          (_aiter5<515000 | _aiter5>515999) &
          (_aiter5<525000 | _aiter5>525999) &
          (_aiter5<535000 | _aiter5>535999) &
          (_aiter5<545000 | _aiter5>545999) &
          (_aiter5<555000 | _aiter5>555999) &
          (_aiter5<565000 | _aiter5>565999) &
          (_aiter5<575000 | _aiter5>575999) &
          (_aiter5<585000 | _aiter5>585999) &
          (_aiter5<595000 | _aiter5>595999))

      _lntotiter=_aiter5+_liter6                          ; FTYPE in order to generate total X for
      _lntotals=_lntotals+_dmiles[_lntotiter]              ; Lanes.

    endif
  ENDLOOP                                                 ;^End Loop 15.

  print list="\\"," ",_lntotals(10.2C)," ", PRINTO=1
  _supertotal=_supertotal+_lntotals                       ;^Generate total X for all ATYPE.

ENDLOOP                                                   ;^End Loop 14.
print list="\\"," ",_supertotal(10.2C), PRINTO=1
print list=" ","\n ", PRINTO=1

;---------------1-DIGIT FACILITY TYPES BY 1-DIGIT AREA TYPES SUMMARY----------------

Print list= "Total Summary Area Types by Facility Types ", PRINTO=1 ;^Header
Print list= "                                                        Single Digit Facility Types
", PRINTO=1
Print list= "AType         1x          2x          3x          4x          5x          6x          7x
8x          9x      Totals", PRINTO=1
Print list= "-------------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1

LOOP _a1iter2=100000,599999,100000                        ;^Begin Loop 16: Cycles through ATYPE by
10 to
  _aat1=int(_a1iter2/100000)                              ; get single digit ATYPE.
  print list= _aat1(1.0),"x","    ", PRINTO=1

  _fttotal=0                                              ;^Initialize total X for all ATYPE

    LOOP _f1iter=1000,9900,1000                           ;^Begin Loop 17: Cycles through FTYPE by
10 to
                                                          ; get single digit FTYPE.
```

```
      _totftlns=0                                              ;^Initialize total X for all FTYPE by
all Lanes.
      if (_f1iter<5000 | _f1iter>5999)
      LOOP _fiter3=_f1iter,9900,100                          ;^Begin Loop 18: Cycles through two-digit
FTYPE
        if (_fiter3>_f1iter+999) BREAK                        ; for current single digit FTYPE in Loop
17.

        LOOP _aiter6=_a1iter2,599999,10000                   ;^Begin Loop 19: Cycles through two-digit
ATYPE
          if (_aiter6>_a1iter2+99999) BREAK                   ; for current single digit ATYPE in Loop
16.

            LOOP _liter7=1,9,1                                          ;^Begin Loop 20: Cycles through
Lanes for current FTYPE and ATYPE
              _totftlns=_totftlns+_dmiles[_aiter6+_fiter3+_liter7]    ; in order to generate total
X for FTYPE by ATYPE.
            ENDLOOP                                                      ;^End Loop 20.

        ENDLOOP                                                 ;^End Loop 19.

      ENDLOOP                                                   ;^End Loop 18.
      endif
      _fttotal=_fttotal+_totftlns                               ;^Generate total X for ATYPE.

      print list="\\"," ",_totftlns(10.2C)," ", PRINTO=1
    ENDLOOP                                                     ;^End Loop 17.

  print list="\\"," ",_fttotal(10.2c), PRINTO=1
ENDLOOP                                                         ;^End Loop 16.

Print list= "-------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1
print list="Totals", PRINTO=1

_supertotal=0                                                   ;^Initialize overall total X.

LOOP _f1iter2=1000,9900,1000                                   ;^Begin Loop 21: Cycles through FTYPE by
10
                                                               ; to get single digit FTYPE.
  _ftotals=0                                                   ;^Initialize total X by FTYPE

  LOOP _fiter4=_f1iter2,9900,100                               ;^Begin Loop 22: Cycles through FTYPE by
1 to
    if (_fiter4>_f1iter2+999) BREAK                             ; get all two-digit FTYPE for current
FTYPE in
    if (_f1iter2<5000 | _f1iter2>5999)                                                    ; Loop
21.
    LOOP _liter8=1,9,1                                         ;^Begin Loop 23: Cycles through Lanes.

      LOOP _aiter7=100000,599999,10000                         ;^Begin Loop 24: Cycles through ATYPE in
order
        _ftotiter=_aiter7+_fiter4+_liter8                       ; to generate total X by single digit
FTYPE.
        _ftotals=_ftotals+_dmiles[_ftotiter]
      ENDLOOP                                                   ;^End Loop 24.

    ENDLOOP                                                     ;^End Loop 23.
    endif
  ENDLOOP                                                       ;^End Loop 22.
  _supertotal=_supertotal+_ftotals                              ;^Generate overall total for all single
digit ATYPE
                                                               ; by all single digit FTYPE.
  print list="\\"," ",_ftotals(10.2C)," ", PRINTO=1
ENDLOOP                                                         ;^End Loop 21.
_dirmiles=_supertotal
print list="\\"," ",_supertotal(10.2C), PRINTO=1
print list=" ", PRINTO=1
;*********************************************************************
; END DIRECTIONAL MILES REPORT
;*********************************************************************
```

```
;=============================================================================================
; BEGIN LANE MILES REPORT ------------ X = LANE MILES
;=============================================================================================
Print list=" ", PRINTO=1
Print
list="*************************************************************************************************
************************************", PRINTO=1
Print                                                                                          list="*
*", PRINTO=1
Print list="*                                          Lane Miles (Centroid Connectors Excluded)
*", PRINTO=1
Print                                                                                          list="*
*", PRINTO=1
Print
list="*************************************************************************************************
************************************", PRINTO=1
Print list=" ", PRINTO=1
;-----------------2-DIGIT FACILITY TYPES BY 2-DIGIT AREA TYPES--------------------

LOOP _a1iter=100000,599999,100000                          ;^Begin Loop 1: Cycles through Area Types
(ATYPE) by 10
  _aat1=int(_a1iter/100000)                                 ; in order to get single digit ATYPE.
  print list= "Area Type ",_aat1(1.0),"x Range:",
           "\n ", PRINTO=1

  LOOP _aiter=_a1iter,599999,10000                          ;^Begin Loop 2: Cycles through ATYPE by
1
    if (_aiter>_a1iter+99999) BREAK                         ; in order to get two-digit ATYPE.
    _aat2=int(_aiter/10000)

    _avcheck=0                                              ;^Initialize ATYPE X checking variable.

    LOOP _achkiter=_aiter,599999,1                          ;^Begin Loop 3: Cycles through Lanes and
Facility Types (FTYPE)
      if (_achkiter>_aiter+9999) BREAK                      ; for current ATYPE in Loop 2 and totals
X checking variable.
      _avcheck=_avcheck+_lmiles[_achkiter]
    ENDLOOP                                                 ;^End Loop 3.

    if (_avcheck>0)                                         ;^Begin Condition 1: If current ATYPE in
Loop 2
                                                            ; has X>0 continue to report X. Else skip
ATYPE.
      _supertotal=0                                         ;^Initialize ATYPE total X.

      Print list= "Area Type ",_aat2(2.0), PRINTO=1         ;^Header
      Print list= "                                                  Number of Lanes per Direction
", PRINTO=1
      Print list= "FType          1           2           3           4           5           6
7           8           9       Totals", PRINTO=1
      Print list= "-----------------------------------------------------------------------------------
-----------------------------------------------", PRINTO=1

      LOOP _fiter=100,9900,100                              ;^Begin Loop 4: Cycles through FTYPE
                                                            ; by 1 in order to get two-digit FTYPE.
       _vcheck=0                                            ;^Initialize FTYPE X checking variable.

       LOOP _liter=1,9,1                                    ;^Begin Loop 5: Cycles through Lanes for
current
         _vcheck=_vcheck+_lmiles[_aiter+_fiter+_liter]     ; FTYPE in Loop 4 and totals X checking
variable.
       ENDLOOP                                              ;^End Loop 5.

        if (_vcheck>0 & (_fiter<5000 | _fiter>5999))        ;^Begin Condition 2: If current FTYPE in
Loop 4
          _fft2=int(_fiter/100)                            ; has X>0 continue to report X. Else skip
FTYPE.
          print list= _fft2(2.0),"    ", PRINTO=1
          _totvols=0                                        ;^Initialize FTYPE total X.
```

```
           LOOP _liter2=1,9,1                              ;^Begin Loop 6: Cycles through Lanes to
generate ATYPE by FTYPE by Lanes total X.
             print list="\\"," ",_lmiles[_aiter+_fiter+_liter2](10.2C)," ", PRINTO=1
             _totvols=_totvols+_lmiles[_aiter+_fiter+_liter2]
             _supertotal=_supertotal+_lmiles[_aiter+_fiter+_liter2]
           ENDLOOP                                         ;^End Loop 6.

             print list="\\"," ",_totvols(10.2C), PRINTO=1
         endif                                             ;^End Condition 2.

       ENDLOOP                                             ;^End Loop 4.

       Print list= "-----------------------------------------------------------------------------
-----------------------------------------------", PRINTO=1
       print list="Totals", PRINTO=1

       LOOP _liter3=1,9,1                                 ;^Begin Loop 7: Cycles through Lanes for
                                                          ; current ATYPE in Loop 2.
         _lntotals=0                                      ;^Initialize Lane total X.

         LOOP _aiter2=_aiter,599999,100                   ;^Begin Loop 8: Cycles through FTYPE for
current ATYPE
           if (_aiter2>_aiter+9999) BREAK                 ; in Loop 2 to generate Lane total X.
           if (_aiter2<_aiter+5000 | _aiter2>_aiter+5999)
             _lntotiter=_aiter2+_liter3
             _lntotals=_lntotals+_lmiles[_lntotiter]
           endif
         ENDLOOP                                          ;^End Loop 8

         print list="\\"," ",_lntotals(10.2C)," ", PRINTO=1
       ENDLOOP                                            ;^End Loop 7

       print list="\\"," ",_supertotal(10.2C), PRINTO=1
       print list=" ", PRINTO=1
     endif                                                ;^End Condition 1.

  ENDLOOP                                                 ;^End Loop 2.

  print list=" ", PRINTO=1
ENDLOOP                                                   ;^End Loop 1.

;-----------------2-DIGIT FACILITY TYPES BY TOTAL AREA TYPES--------------------

Print list= "Total Area Types ", PRINTO=1                 ;^Header
Print list= "                                                    Number of Lanes per Direction
", PRINTO=1
Print list= "FType         1          2          3          4          5          6          7
8          9       Totals", PRINTO=1
Print list= "-----------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1

LOOP _fiter2=100,9900,100                                 ;^Begin Loop 9: Cycles through FTYPES to
get
  _fft2=int(_fiter2/100)                                  ; two-digit FTYPE.

  _tafvcheck=0                                            ;^Initialize FTYPE X checking variable.
 if (_fft2<50 | _fft2>59)
  LOOP _liter5=1,9,1                                        ;^Begin Loop 10: Cycles through Lanes
for current
                                                          ; FTYPE in Loop 9.
    LOOP _aiter4= 100000,599999,10000                       ;^Begin Loop 11: Cycles through ATYPE
for
      _tafvcheck=_tafvcheck+_lmiles[_aiter4+_fiter2+_liter5] ; current Lanes and FTYPE in order to
total X checking variable.
    ENDLOOP                                                 ;^End Loop 11.

  ENDLOOP                                                 ;^End Loop 10.

  if (_tafvcheck>0)                                       ;^Begin Condition 3: If current FTYPE in
Loop 9
```

```
    print list= _fft2(2.0),"    ", PRINTO=1                ; has X>0 continue to report X. Else skip
FTYPE.

    LOOP _liter4= 1,9,1                                      ;^Begin Loop 12: Cycles through Lanes
for current FTYPE
                                                            ; in Loop 9.
        _totftat=0                                         ;^Initialize FTYPE total X for all ATYPE.

      LOOP _aiter3= 100000,599999,10000                      ;^Begin Loop 13: Cycles through ATYPE
for current Lanes in Loop 12
        _totftat=_totftat+_lmiles[_aiter3+_fiter2+_liter4] ; in order to generate total X for FTYPE
by Lane for all ATYPE.
      ENDLOOP                                               ;^End Loop 13.

      print list="\\"," ",_totftat(10.2C)," ", PRINTO=1
    ENDLOOP                                                 ;^End Loop 12.

    print list="\\"," ",_tafvcheck(10.2C), PRINTO=1
  endif                                                     ;^End Condition 3.
 endif
ENDLOOP                                                     ;^End Loop 9.

Print list= "-----------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1
print list="Totals", PRINTO=1

_supertotal=0                                               ;^Initialize all ATYPE total X.

LOOP _liter6=1,9,1                                          ;^Begin Loop 14: Cycles through Lanes.

  _lntotals=0                                               ;^Initialize total X for Lanes.

  LOOP _aiter5=100000,599999,100                              ;^Begin Loop 15: Cycles through ATYPE
and
    if ((_aiter5<105000 | _aiter5>105999) &
        (_aiter5<115000 | _aiter5>115999) &
        (_aiter5<125000 | _aiter5>125999) &
        (_aiter5<135000 | _aiter5>135999) &
        (_aiter5<145000 | _aiter5>145999) &
        (_aiter5<155000 | _aiter5>155999) &
        (_aiter5<165000 | _aiter5>165999) &
        (_aiter5<175000 | _aiter5>175999) &
        (_aiter5<185000 | _aiter5>185999) &
        (_aiter5<195000 | _aiter5>195999) &
        (_aiter5<205000 | _aiter5>205999) &
        (_aiter5<215000 | _aiter5>215999) &
        (_aiter5<225000 | _aiter5>225999) &
        (_aiter5<235000 | _aiter5>235999) &
        (_aiter5<245000 | _aiter5>245999) &
        (_aiter5<255000 | _aiter5>255999) &
        (_aiter5<265000 | _aiter5>265999) &
        (_aiter5<275000 | _aiter5>275999) &
        (_aiter5<285000 | _aiter5>285999) &
        (_aiter5<295000 | _aiter5>295999) &
        (_aiter5<305000 | _aiter5>305999) &
        (_aiter5<315000 | _aiter5>315999) &
        (_aiter5<325000 | _aiter5>325999) &
        (_aiter5<335000 | _aiter5>335999) &
        (_aiter5<345000 | _aiter5>345999) &
        (_aiter5<355000 | _aiter5>355999) &
        (_aiter5<365000 | _aiter5>365999) &
        (_aiter5<375000 | _aiter5>375999) &
        (_aiter5<385000 | _aiter5>385999) &
        (_aiter5<395000 | _aiter5>395999) &
        (_aiter5<405000 | _aiter5>405999) &
        (_aiter5<415000 | _aiter5>415999) &
        (_aiter5<425000 | _aiter5>425999) &
        (_aiter5<435000 | _aiter5>435999) &
        (_aiter5<445000 | _aiter5>445999) &
        (_aiter5<455000 | _aiter5>455999) &
        (_aiter5<465000 | _aiter5>465999) &
```

```
        (_aiter5<475000 | _aiter5>475999) &
        (_aiter5<485000 | _aiter5>485999) &
        (_aiter5<495000 | _aiter5>495999) &
        (_aiter5<505000 | _aiter5>505999) &
        (_aiter5<515000 | _aiter5>515999) &
        (_aiter5<525000 | _aiter5>525999) &
        (_aiter5<535000 | _aiter5>535999) &
        (_aiter5<545000 | _aiter5>545999) &
        (_aiter5<555000 | _aiter5>555999) &
        (_aiter5<565000 | _aiter5>565999) &
        (_aiter5<575000 | _aiter5>575999) &
        (_aiter5<585000 | _aiter5>585999) &
        (_aiter5<595000 | _aiter5>595999))

      _lntotiter=_aiter5+_liter6                          ; FTYPE in order to generate total X for
      _lntotals=_lntotals+_lmiles[_lntotiter]              ; Lanes.

    endif
  ENDLOOP                                                 ;^End Loop 15.

  print list="\\"," ",_lntotals(10.2C)," ", PRINTO=1
  _supertotal=_supertotal+_lntotals                       ;^Generate total X for all ATYPE.

ENDLOOP                                                   ;^End Loop 14.
print list="\\"," ",_supertotal(10.2C), PRINTO=1
print list=" ","\n ", PRINTO=1

;---------------1-DIGIT FACILITY TYPES BY 1-DIGIT AREA TYPES SUMMARY----------------

Print list= "Total Summary Area Types by Facility Types ", PRINTO=1 ;^Header
Print list= "                                                    Single Digit Facility Types
", PRINTO=1
Print list= "AType         1x          2x          3x          4x          5x          6x          7x
8x          9x      Totals", PRINTO=1
Print list= "-------------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1

LOOP _a1iter2=100000,599999,100000                        ;^Begin Loop 16: Cycles through ATYPE by
10 to
  _aat1=int(_a1iter2/100000)                              ; get single digit ATYPE.
  print list= _aat1(1.0),"x","    ", PRINTO=1

  _fttotal=0                                              ;^Initialize total X for all ATYPE

    LOOP _f1iter=1000,9900,1000                           ;^Begin Loop 17: Cycles through FTYPE by
10 to
                                                          ; get single digit FTYPE.
      _totftlns=0                                           ;^Initialize total X for all FTYPE by
all Lanes.
      if (_f1iter<5000 | _f1iter>5999)
      LOOP _fiter3=_f1iter,9900,100                       ;^Begin Loop 18: Cycles through two-digit
FTYPE
        if (_fiter3>_f1iter+999) BREAK                    ; for current single digit FTYPE in Loop
17.

        LOOP _aiter6=_a1iter2,599999,10000                ;^Begin Loop 19: Cycles through two-digit
ATYPE
          if (_aiter6>_a1iter2+99999) BREAK               ; for current single digit ATYPE in Loop
16.

            LOOP _liter7=1,9,1                                ;^Begin Loop 20: Cycles through
Lanes for current FTYPE and ATYPE
              _totftlns=_totftlns+_lmiles[_aiter6+_fiter3+_liter7]   ; in order to generate total
X for FTYPE by ATYPE.
            ENDLOOP                                             ;^End Loop 20.

        ENDLOOP                                           ;^End Loop 19.

      ENDLOOP                                             ;^End Loop 18.
      endif
      _fttotal=_fttotal+_totftlns                         ;^Generate total X for ATYPE.
```

```
      print list="\\"," ",_totftlns(10.2C)," ", PRINTO=1
    ENDLOOP                                                ;^End Loop 17.

  print list="\\"," ",_fttotal(10.2c), PRINTO=1
ENDLOOP                                                    ;^End Loop 16.

Print list= "-------------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1
print list="Totals", PRINTO=1

_supertotal=0                                              ;^Initialize overall total X.

LOOP _f1iter2=1000,9900,1000                               ;^Begin Loop 21: Cycles through FTYPE by
10
                                                           ; to get single digit FTYPE.
  _ftotals=0                                               ;^Initialize total X by FTYPE

  LOOP _fiter4=_f1iter2,9900,100                           ;^Begin Loop 22: Cycles through FTYPE by
1 to
    if (_fiter4>_f1iter2+999) BREAK                          ; get all two-digit FTYPE for current
FTYPE in
    if (_f1iter2<5000 | _f1iter2>5999)                                                  ; Loop
21.
    LOOP _liter8=1,9,1                                     ;^Begin Loop 23: Cycles through Lanes.

      LOOP _aiter7=100000,599999,10000                     ;^Begin Loop 24: Cycles through ATYPE in
order
        _ftotiter=_aiter7+_fiter4+_liter8                    ; to generate total X by single digit
FTYPE.
        _ftotals=_ftotals+_lmiles[_ftotiter]
      ENDLOOP                                              ;^End Loop 24.

    ENDLOOP                                                ;^End Loop 23.
    endif
  ENDLOOP                                                  ;^End Loop 22.
  _supertotal=_supertotal+_ftotals                         ;^Generate overall total for all single
digit ATYPE
                                                           ; by all single digit FTYPE.
  print list="\\"," ",_ftotals(10.2C)," ", PRINTO=1
ENDLOOP                                                    ;^End Loop 21.
_lanemiles=_supertotal
print list="\\"," ",_supertotal(10.2C), PRINTO=1
print list=" ", PRINTO=1
;*************************************************************************
; END LANE MILES REPORT
;*************************************************************************

;=========================================================================================
; BEGIN VMT VAL REPORT ------------ X = VMT on Links w/ Counts
;=========================================================================================
Print list=" ", PRINTO=1
Print
list="*****************************************************************************************
***********************************", PRINTO=1
Print                                                                                 list="*
*", PRINTO=1
Print list="*                                  Vehicle Miles Traveled (VMT) using Volumes on Links
with Counts                                *", PRINTO=1
Print                                                                                 list="*
*", PRINTO=1
Print
list="*****************************************************************************************
***********************************", PRINTO=1
Print list=" ", PRINTO=1
;-----------------2-DIGIT FACILITY TYPES BY 2-DIGIT AREA TYPES--------------------

LOOP _a1iter=100000,599999,100000                          ;^Begin Loop 1: Cycles through Area Types
(ATYPE) by 10
  _aat1=int(_a1iter/100000)                                 ; in order to get single digit ATYPE.
  print list= "Area Type ",_aat1(1.0),"x Range:",
```

```
          "\n ", PRINTO=1

  LOOP _aiter=_a1iter,599999,10000                          ;^Begin Loop 2: Cycles through ATYPE by
1
    if (_aiter>_a1iter+99999) BREAK                         ; in order to get two-digit ATYPE.
    _aat2=int(_aiter/10000)

    _avcheck=0                                              ;^Initialize ATYPE X checking variable.

    LOOP _achkiter=_aiter,599999,1                          ;^Begin Loop 3: Cycles through Lanes and
Facility Types (FTYPE)
      if (_achkiter>_aiter+9999) BREAK                      ; for current ATYPE in Loop 2 and totals
X checking variable.
      _avcheck=_avcheck+_volvmtval[_achkiter]
    ENDLOOP                                                 ;^End Loop 3.

    if (_avcheck>0)                                         ;^Begin Condition 1: If current ATYPE in
Loop 2
                                                            ; has X>0 continue to report X. Else skip
ATYPE.
      _supertotal=0                                         ;^Initialize ATYPE total X.

      Print list= "Area Type ",_aat2(2.0), PRINTO=1         ;^Header
      Print list= "                                                  Number of Lanes per Direction
", PRINTO=1
      Print list= "FType          1           2           3           4           5           6
7           8           9       Totals", PRINTO=1
      Print list= "-------------------------------------------------------------------------------
-----------------------------------------------", PRINTO=1

      LOOP _fiter=100,9900,100                              ;^Begin Loop 4: Cycles through FTYPE
                                                            ; by 1 in order to get two-digit FTYPE.
       _vcheck=0                                            ;^Initialize FTYPE X checking variable.

       LOOP _liter=1,9,1                                    ;^Begin Loop 5: Cycles through Lanes for
current
         _vcheck=_vcheck+_volvmtval[_aiter+_fiter+_liter]      ; FTYPE in Loop 4 and totals X
checking variable.
       ENDLOOP                                              ;^End Loop 5.

        if (_vcheck>0)                                      ;^Begin Condition 2: If current FTYPE in
Loop 4
          _fft2=int(_fiter/100)                            ; has X>0 continue to report X. Else skip
FTYPE.
          print list= _fft2(2.0),"    ", PRINTO=1
          _totvols=0                                        ;^Initialize FTYPE total X.

          LOOP _liter2=1,9,1                                ;^Begin Loop 6: Cycles through Lanes to
generate ATYPE by FTYPE by Lanes total X.
            print list="\\"," ",_volvmtval[_aiter+_fiter+_liter2](10.0C)," ", PRINTO=1
            _totvols=_totvols+_volvmtval[_aiter+_fiter+_liter2]
            _supertotal=_supertotal+_volvmtval[_aiter+_fiter+_liter2]
          ENDLOOP                                           ;^End Loop 6.

            print list="\\"," ",_totvols(10.0C), PRINTO=1
        endif                                               ;^End Condition 2.

      ENDLOOP                                               ;^End Loop 4.

      Print list= "-------------------------------------------------------------------------------
-----------------------------------------------", PRINTO=1
      print list="Totals", PRINTO=1

      LOOP _liter3=1,9,1                                    ;^Begin Loop 7: Cycles through Lanes for
                                                            ; current ATYPE in Loop 2.
        _lntotals=0                                         ;^Initialize Lane total X.

        LOOP _aiter2=_aiter,599999,100                      ;^Begin Loop 8: Cycles through FTYPE for
current ATYPE
          if (_aiter2>_aiter+9999) BREAK                    ; in Loop 2 to generate Lane total X.
          _lntotiter=_aiter2+_liter3
```

```
          _lntotals=_lntotals+_volvmtval[_lntotiter]
        ENDLOOP                                                ;^End Loop 8

        print list="\\"," ",_lntotals(10.0C)," ", PRINTO=1
      ENDLOOP                                                  ;^End Loop 7

      print list="\\"," ",_supertotal(10.0C), PRINTO=1
      print list=" ", PRINTO=1
    endif                                                      ;^End Condition 1.

  ENDLOOP                                                      ;^End Loop 2.

  print list=" ", PRINTO=1
ENDLOOP                                                        ;^End Loop 1.

;-----------------2-DIGIT FACILITY TYPES BY TOTAL AREA TYPES--------------------

Print list= "Total Area Types ", PRINTO=1                      ;^Header
Print list= "                                                          Number of Lanes per Direction
", PRINTO=1
Print list= "FType         1           2           3           4           5           6           7
8           9       Totals", PRINTO=1
Print list= "-------------------------------------------------------------------------------------------
-----------------------------------------", PRINTO=1

LOOP _fiter2=100,9900,100                                     ;^Begin Loop 9: Cycles through FTYPES to
get
  _fft2=int(_fiter2/100)                                       ; two-digit FTYPE.

  _tafvcheck=0                                                 ;^Initialize FTYPE X checking variable.

  LOOP _liter5=1,9,1                                             ;^Begin Loop 10: Cycles through Lanes
for current
                                                               ; FTYPE in Loop 9.
    LOOP _aiter4= 100000,599999,10000                            ;^Begin Loop 11: Cycles through ATYPE
for
      _tafvcheck=_tafvcheck+_volvmtval[_aiter4+_fiter2+_liter5] ; current Lanes and FTYPE in order
to total X checking variable.
    ENDLOOP                                                      ;^End Loop 11.

  ENDLOOP                                                      ;^End Loop 10.

  if (_tafvcheck>0)                                            ;^Begin Condition 3: If current FTYPE in
Loop 9
    print list= _fft2(2.0),"    ", PRINTO=1                    ; has X>0 continue to report X. Else skip
FTYPE.

    LOOP _liter4= 1,9,1                                          ;^Begin Loop 12: Cycles through Lanes
for current FTYPE
                                                               ; in Loop 9.
      _totftat=0                                               ;^Initialize FTYPE total X for all ATYPE.

      LOOP _aiter3= 100000,599999,10000                          ;^Begin Loop 13: Cycles through ATYPE
for current Lanes in Loop 12
        _totftat=_totftat+_volvmtval[_aiter3+_fiter2+_liter4] ; in order to generate total X for
FTYPE by Lane for all ATYPE.
      ENDLOOP                                                  ;^End Loop 13.

      print list="\\"," ",_totftat(10.0C)," ", PRINTO=1
    ENDLOOP                                                    ;^End Loop 12.

    print list="\\"," ",_tafvcheck(10.0C), PRINTO=1
  endif                                                        ;^End Condition 3.

ENDLOOP                                                        ;^End Loop 9.

Print list= "-------------------------------------------------------------------------------------------
-----------------------------------------", PRINTO=1
print list="Totals", PRINTO=1

_supertotal=0                                                  ;^Initialize all ATYPE total X.
```

```
LOOP _liter6=1,9,1                                          ;^Begin Loop 14: Cycles through Lanes.

  _lntotals=0                                               ;^Initialize total X for Lanes.

  LOOP _aiter5=100000,599999,100                              ;^Begin Loop 15: Cycles through ATYPE
and
      _lntotiter=_aiter5+_liter6                            ; FTYPE in order to generate total X for
      _lntotals=_lntotals+_volvmtval[_lntotiter]                ; Lanes.
  ENDLOOP                                                   ;^End Loop 15.

  print list="\\"," ",_lntotals(10.0C)," ", PRINTO=1
  _supertotal=_supertotal+_lntotals                         ;^Generate total X for all ATYPE.

ENDLOOP                                                     ;^End Loop 14.
print list="\\"," ",_supertotal(10.0C), PRINTO=1
print list=" ","\n ", PRINTO=1

;---------------1-DIGIT FACILITY TYPES BY 1-DIGIT AREA TYPES SUMMARY----------------

Print list= "Total Summary Area Types by Facility Types ", PRINTO=1 ;^Header
Print list= "                                                        Single Digit Facility Types
", PRINTO=1
Print list= "AType         1x          2x          3x          4x          5x          6x          7x
8x          9x      Totals", PRINTO=1
Print list= "-------------------------------------------------------------------------------------------
-----------------------------------------", PRINTO=1

LOOP _a1iter2=100000,599999,100000                          ;^Begin Loop 16: Cycles through ATYPE by
10 to
  _aat1=int(_a1iter2/100000)                                ; get single digit ATYPE.
  print list= _aat1(1.0),"x","    ", PRINTO=1

  _fttotal=0                                                ;^Initialize total X for all ATYPE

    LOOP _f1iter=1000,9900,1000                             ;^Begin Loop 17: Cycles through FTYPE by
10 to
                                                            ; get single digit FTYPE.
      _totftlns=0                                             ;^Initialize total X for all FTYPE by
all Lanes.

      LOOP _fiter3=_f1iter,9900,100                         ;^Begin Loop 18: Cycles through two-digit
FTYPE
        if (_fiter3>_f1iter+999) BREAK                      ; for current single digit FTYPE in Loop
17.

        LOOP _aiter6=_a1iter2,599999,10000                  ;^Begin Loop 19: Cycles through two-digit
ATYPE
          if (_aiter6>_a1iter2+99999) BREAK                 ; for current single digit ATYPE in Loop
16.

            LOOP _liter7=1,9,1                                      ;^Begin Loop 20: Cycles through
Lanes for current FTYPE and ATYPE
              _totftlns=_totftlns+_volvmtval[_aiter6+_fiter3+_liter7]    ; in order to generate
total X for FTYPE by ATYPE.
            ENDLOOP                                                   ;^End Loop 20.

        ENDLOOP                                             ;^End Loop 19.

      ENDLOOP                                               ;^End Loop 18.

      _fttotal=_fttotal+_totftlns                           ;^Generate total X for ATYPE.

      print list="\\"," ",_totftlns(10.0C)," ", PRINTO=1
    ENDLOOP                                                 ;^End Loop 17.

  print list="\\"," ",_fttotal(10.0c), PRINTO=1
ENDLOOP                                                     ;^End Loop 16.

Print list= "-------------------------------------------------------------------------------------------
-----------------------------------------", PRINTO=1
```

```
print list="Totals", PRINTO=1

_supertotal=0                                                ;^Initialize overall total X.

LOOP _f1iter2=1000,9900,1000                                 ;^Begin Loop 21: Cycles through FTYPE by
10
                                                             ; to get single digit FTYPE.
  _ftotals=0                                                 ;^Initialize total X by FTYPE

  LOOP _fiter4=_f1iter2,9900,100                             ;^Begin Loop 22: Cycles through FTYPE by
1 to
    if (_fiter4>_f1iter2+999) BREAK                            ; get all two-digit FTYPE for current
FTYPE in
                                                             ; Loop 21.
    LOOP _liter8=1,9,1                                       ;^Begin Loop 23: Cycles through Lanes.

      LOOP _aiter7=100000,599999,10000                       ;^Begin Loop 24: Cycles through ATYPE in
order
        _ftotiter=_aiter7+_fiter4+_liter8                      ; to generate total X by single digit
FTYPE.
        _ftotals=_ftotals+_volvmtval[_ftotiter]
      ENDLOOP                                                ;^End Loop 24.

    ENDLOOP                                                  ;^End Loop 23.

  ENDLOOP                                                    ;^End Loop 22.

  _supertotal=_supertotal+_ftotals                           ;^Generate overall total for all single
digit ATYPE
                                                             ; by all single digit FTYPE.
  print list="\\"," ",_ftotals(10.0C)," ", PRINTO=1
ENDLOOP                                                      ;^End Loop 21.
_vmtvoloncounts=_supertotal
print list="\\"," ",_supertotal(10.0C), PRINTO=1
print list=" ", PRINTO=1
;*************************************************************************
; END VMT VAL REPORT
;*************************************************************************

;=========================================================================================
; BEGIN VMT Count REPORT ------------ X = Count VMT on Links w/ Counts
;=========================================================================================
Print list=" ", PRINTO=1
Print
list="*****************************************************************************************
***********************************", PRINTO=1
Print                                                                                  list="*
*", PRINTO=1
Print list="*                                 Vehicle Miles Traveled (VMT) using Counts on Links with
Counts                                  *", PRINTO=1
Print                                                                                  list="*
*", PRINTO=1
Print
list="*****************************************************************************************
***********************************", PRINTO=1
Print list=" ", PRINTO=1
;-----------------2-DIGIT FACILITY TYPES BY 2-DIGIT AREA TYPES--------------------

LOOP _a1iter=100000,599999,100000                           ;^Begin Loop 1: Cycles through Area Types
(ATYPE) by 10
  _aat1=int(_a1iter/100000)                                  ; in order to get single digit ATYPE.
  print list= "Area Type ",_aat1(1.0),"x Range:",
           "\n ", PRINTO=1

  LOOP _aiter=_a1iter,599999,10000                           ;^Begin Loop 2: Cycles through ATYPE by
1
    if (_aiter>_a1iter+99999) BREAK                          ; in order to get two-digit ATYPE.
    _aat2=int(_aiter/10000)


    _avcheck=0                                               ;^Initialize ATYPE X checking variable.
```

```
    LOOP _achkiter=_aiter,599999,1                          ;^Begin Loop 3: Cycles through Lanes and
Facility Types (FTYPE)
      if (_achkiter>_aiter+9999) BREAK                      ; for current ATYPE in Loop 2 and totals
X checking variable.
      _avcheck=_avcheck+_cntvmtval[_achkiter]
    ENDLOOP                                                 ;^End Loop 3.

    if (_avcheck>0)                                         ;^Begin Condition 1: If current ATYPE in
Loop 2
                                                            ; has X>0 continue to report X. Else skip
ATYPE.
      _supertotal=0                                         ;^Initialize ATYPE total X.

      Print list= "Area Type ",_aat2(2.0), PRINTO=1         ;^Header
      Print list= "                                                    Number of Lanes per Direction
", PRINTO=1
      Print list= "FType          1           2           3           4           5           6
7           8           9       Totals", PRINTO=1
      Print list= "-------------------------------------------------------------------------------
-----------------------------------------------", PRINTO=1

      LOOP _fiter=100,9900,100                              ;^Begin Loop 4: Cycles through FTYPE
                                                            ; by 1 in order to get two-digit FTYPE.
       _vcheck=0                                            ;^Initialize FTYPE X checking variable.

       LOOP _liter=1,9,1                                    ;^Begin Loop 5: Cycles through Lanes for
current
         _vcheck=_vcheck+_cntvmtval[_aiter+_fiter+_liter]      ; FTYPE in Loop 4 and totals X
checking variable.
       ENDLOOP                                              ;^End Loop 5.

        if (_vcheck>0)                                      ;^Begin Condition 2: If current FTYPE in
Loop 4
          _fft2=int(_fiter/100)                            ; has X>0 continue to report X. Else skip
FTYPE.
          print list= _fft2(2.0),"    ", PRINTO=1
          _totvols=0                                        ;^Initialize FTYPE total X.

          LOOP _liter2=1,9,1                                ;^Begin Loop 6: Cycles through Lanes to
generate ATYPE by FTYPE by Lanes total X.
            print list="\\"," ",_cntvmtval[_aiter+_fiter+_liter2](10.0C)," ", PRINTO=1
            _totvols=_totvols+_cntvmtval[_aiter+_fiter+_liter2]
            _supertotal=_supertotal+_cntvmtval[_aiter+_fiter+_liter2]
          ENDLOOP                                           ;^End Loop 6.

            print list="\\"," ",_totvols(10.0C), PRINTO=1
        endif                                               ;^End Condition 2.

      ENDLOOP                                               ;^End Loop 4.

      Print list= "-------------------------------------------------------------------------------
-----------------------------------------------", PRINTO=1
      print list="Totals", PRINTO=1

      LOOP _liter3=1,9,1                                    ;^Begin Loop 7: Cycles through Lanes for
                                                            ; current ATYPE in Loop 2.
        _lntotals=0                                         ;^Initialize Lane total X.

        LOOP _aiter2=_aiter,599999,100                      ;^Begin Loop 8: Cycles through FTYPE for
current ATYPE
          if (_aiter2>_aiter+9999) BREAK                    ; in Loop 2 to generate Lane total X.
          _lntotiter=_aiter2+_liter3
          _lntotals=_lntotals+_cntvmtval[_lntotiter]
        ENDLOOP                                             ;^End Loop 8

        print list="\\"," ",_lntotals(10.0C)," ", PRINTO=1
      ENDLOOP                                               ;^End Loop 7

      print list="\\"," ",_supertotal(10.0C), PRINTO=1
      print list=" ", PRINTO=1
```

```
    endif                                                    ;^End Condition 1.

  ENDLOOP                                                    ;^End Loop 2.

  print list=" ", PRINTO=1
ENDLOOP                                                      ;^End Loop 1.

;-----------------2-DIGIT FACILITY TYPES BY TOTAL AREA TYPES--------------------

Print list= "Total Area Types ", PRINTO=1                    ;^Header
Print list= "                                                        Number of Lanes per Direction
", PRINTO=1
Print list= "FType         1           2           3           4           5           6           7
8           9       Totals", PRINTO=1
Print list= "-------------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1

LOOP _fiter2=100,9900,100                                   ;^Begin Loop 9: Cycles through FTYPES to
get
  _fft2=int(_fiter2/100)                                     ; two-digit FTYPE.

  _tafvcheck=0                                               ;^Initialize FTYPE X checking variable.

  LOOP _liter5=1,9,1                                           ;^Begin Loop 10: Cycles through Lanes
for current
                                                             ; FTYPE in Loop 9.
    LOOP _aiter4= 100000,599999,10000                          ;^Begin Loop 11: Cycles through ATYPE
for
      _tafvcheck=_tafvcheck+_cntvmtval[_aiter4+_fiter2+_liter5] ; current Lanes and FTYPE in order
to total X checking variable.
    ENDLOOP                                                    ;^End Loop 11.

  ENDLOOP                                                    ;^End Loop 10.

  if (_tafvcheck>0)                                          ;^Begin Condition 3: If current FTYPE in
Loop 9
    print list= _fft2(2.0),"    ", PRINTO=1                  ; has X>0 continue to report X. Else skip
FTYPE.

    LOOP _liter4= 1,9,1                                        ;^Begin Loop 12: Cycles through Lanes
for current FTYPE
                                                             ; in Loop 9.
      _totftat=0                                            ;^Initialize FTYPE total X for all ATYPE.

      LOOP _aiter3= 100000,599999,10000                        ;^Begin Loop 13: Cycles through ATYPE
for current Lanes in Loop 12
        _totftat=_totftat+_cntvmtval[_aiter3+_fiter2+_liter4] ; in order to generate total X for
FTYPE by Lane for all ATYPE.
      ENDLOOP                                                ;^End Loop 13.

      print list="\\"," ",_totftat(10.0C)," ", PRINTO=1
    ENDLOOP                                                  ;^End Loop 12.

    print list="\\"," ",_tafvcheck(10.0C), PRINTO=1
  endif                                                      ;^End Condition 3.

ENDLOOP                                                      ;^End Loop 9.

Print list= "-------------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1
print list="Totals", PRINTO=1

_supertotal=0                                                ;^Initialize all ATYPE total X.

LOOP _liter6=1,9,1                                           ;^Begin Loop 14: Cycles through Lanes.

  _lntotals=0                                                ;^Initialize total X for Lanes.

  LOOP _aiter5=100000,599999,100                               ;^Begin Loop 15: Cycles through ATYPE
and
      _lntotiter=_aiter5+_liter6                             ; FTYPE in order to generate total X for
```

```
      _lntotals=_lntotals+_cntvmtval[_lntotiter]              ; Lanes.
  ENDLOOP                                                  ;^End Loop 15.

  print list="\\"," ",_lntotals(10.0C)," ", PRINTO=1
  _supertotal=_supertotal+_lntotals                        ;^Generate total X for all ATYPE.

ENDLOOP                                                    ;^End Loop 14.
print list="\\"," ",_supertotal(10.0C), PRINTO=1
print list=" ","\n ", PRINTO=1

;---------------1-DIGIT FACILITY TYPES BY 1-DIGIT AREA TYPES SUMMARY----------------

Print list= "Total Summary Area Types by Facility Types ", PRINTO=1 ;^Header
Print list= "                                                        Single Digit Facility Types
", PRINTO=1
Print list= "AType         1x          2x          3x          4x          5x          6x          7x
8x          9x      Totals", PRINTO=1
Print list= "-------------------------------------------------------------------------------------------
-----------------------------------------", PRINTO=1

LOOP _a1iter2=100000,599999,100000                         ;^Begin Loop 16: Cycles through ATYPE by
10 to
  _aat1=int(_a1iter2/100000)                               ; get single digit ATYPE.
  print list= _aat1(1.0),"x","    ", PRINTO=1

  _fttotal=0                                               ;^Initialize total X for all ATYPE

    LOOP _f1iter=1000,9900,1000                            ;^Begin Loop 17: Cycles through FTYPE by
10 to
                                                           ; get single digit FTYPE.
      _totftlns=0                                            ;^Initialize total X for all FTYPE by
all Lanes.

      LOOP _fiter3=_f1iter,9900,100                        ;^Begin Loop 18: Cycles through two-digit
FTYPE
        if (_fiter3>_f1iter+999) BREAK                     ; for current single digit FTYPE in Loop
17.

        LOOP _aiter6=_a1iter2,599999,10000                 ;^Begin Loop 19: Cycles through two-digit
ATYPE
          if (_aiter6>_a1iter2+99999) BREAK                ; for current single digit ATYPE in Loop
16.

            LOOP _liter7=1,9,1                                    ;^Begin Loop 20: Cycles through
Lanes for current FTYPE and ATYPE
              _totftlns=_totftlns+_cntvmtval[_aiter6+_fiter3+_liter7]     ; in order to generate
total X for FTYPE by ATYPE.
            ENDLOOP                                                 ;^End Loop 20.

        ENDLOOP                                            ;^End Loop 19.

      ENDLOOP                                              ;^End Loop 18.

      _fttotal=_fttotal+_totftlns                          ;^Generate total X for ATYPE.

      print list="\\"," ",_totftlns(10.0C)," ", PRINTO=1
    ENDLOOP                                                ;^End Loop 17.

  print list="\\"," ",_fttotal(10.0c), PRINTO=1
ENDLOOP                                                    ;^End Loop 16.

Print list= "-------------------------------------------------------------------------------------------
-----------------------------------------", PRINTO=1
print list="Totals", PRINTO=1

_supertotal=0                                              ;^Initialize overall total X.

LOOP _f1iter2=1000,9900,1000                               ;^Begin Loop 21: Cycles through FTYPE by
10
                                                           ; to get single digit FTYPE.
  _ftotals=0                                               ;^Initialize total X by FTYPE
```

```
  LOOP _fiter4=_f1iter2,9900,100                          ;^Begin Loop 22: Cycles through FTYPE by
1 to
    if (_fiter4>_f1iter2+999) BREAK                         ; get all two-digit FTYPE for current
FTYPE in
                                                            ; Loop 21.
    LOOP _liter8=1,9,1                                      ;^Begin Loop 23: Cycles through Lanes.

      LOOP _aiter7=100000,599999,10000                      ;^Begin Loop 24: Cycles through ATYPE in
order
        _ftotiter=_aiter7+_fiter4+_liter8                   ; to generate total X by single digit
FTYPE.
        _ftotals=_ftotals+_cntvmtval[_ftotiter]
      ENDLOOP                                               ;^End Loop 24.

    ENDLOOP                                                 ;^End Loop 23.

  ENDLOOP                                                   ;^End Loop 22.

  _supertotal=_supertotal+_ftotals                          ;^Generate overall total for all single
digit ATYPE
                                                            ; by all single digit FTYPE.
  print list="\\"," ",_ftotals(10.0C)," ", PRINTO=1
ENDLOOP                                                     ;^End Loop 21.
_vmtcountsoncounts=_supertotal
print list="\\"," ",_supertotal(10.0C), PRINTO=1
print list=" ", PRINTO=1
;*********************************************************************
; END COUNT VMT REPORT
;*********************************************************************

;=============================================================================================
; BEGIN VOLUME/COUNT VMT REPORT ------------ X = Volumes over Counts VMT on Links w/ Counts
;=============================================================================================
Print list=" ", PRINTO=1
Print
list="*************************************************************************************
**********************************", PRINTO=1
Print                                                                                  list="*
*", PRINTO=1
Print list="*                                  VMT Volume over Count Ratios on Links with Counts
*", PRINTO=1
Print                                                                                  list="*
*", PRINTO=1
Print
list="*************************************************************************************
**********************************", PRINTO=1
Print list=" ", PRINTO=1
;-----------------2-DIGIT FACILITY TYPES BY 2-DIGIT AREA TYPES--------------------

LOOP _a1iter=100000,599999,100000                          ;^Begin Loop 1: Cycles through Area Types
(ATYPE) by 10
  _aat1=int(_a1iter/100000)                                 ; in order to get single digit ATYPE.
  print list= "Area Type ",_aat1(1.0),"x Range:",
          "\n ", PRINTO=1

  LOOP _aiter=_a1iter,599999,10000                          ;^Begin Loop 2: Cycles through ATYPE by
1
    if (_aiter>_a1iter+99999) BREAK                         ; in order to get two-digit ATYPE.
    _aat2=int(_aiter/10000)

    _avcheck=0                                              ;^Initialize ATYPE X checking variable.

    LOOP _achkiter=_aiter,599999,1                          ;^Begin Loop 3: Cycles through Lanes and
Facility Types (FTYPE)
      if (_achkiter>_aiter+9999) BREAK                      ; for current ATYPE in Loop 2 and totals
X checking variable.
      _avcheck=_avcheck+_vcntby[_achkiter]
    ENDLOOP                                                 ;^End Loop 3.
```

```
    if (_avcheck>0)                                          ;^Begin Condition 1: If current ATYPE in
Loop 2
                                                             ; has X>0 continue to report X. Else skip
ATYPE.
      _supertotal=0                                          ;^Initialize ATYPE total X.
      _supercnts=0

      Print list= "Area Type ",_aat2(2.0), PRINTO=1          ;^Header
      Print list= "                                                    Number of Lanes per Direction
", PRINTO=1
      Print list= "FType          1           2           3           4           5           6
7           8           9       Totals", PRINTO=1
      Print list= "-------------------------------------------------------------------------------
-----------------------------------------------", PRINTO=1

      LOOP _fiter=100,9900,100                               ;^Begin Loop 4: Cycles through FTYPE
                                                             ; by 1 in order to get two-digit FTYPE.
       _vcheck=0                                             ;^Initialize FTYPE X checking variable.

       LOOP _liter=1,9,1                                     ;^Begin Loop 5: Cycles through Lanes for
current
         _vcheck=_vcheck+_vcntby[_aiter+_fiter+_liter]     ; FTYPE in Loop 4 and totals X checking
variable.
       ENDLOOP                                               ;^End Loop 5.

        if (_vcheck>0)                                       ;^Begin Condition 2: If current FTYPE in
Loop 4
          _fft2=int(_fiter/100)                             ; has X>0 continue to report X. Else skip
FTYPE.
          print list= _fft2(2.0),"    ", PRINTO=1
          _totvols=0                                         ;^Initialize FTYPE total X.
          _totcnts=0

           LOOP _liter2=1,9,1                                ;^Begin Loop 6: Cycles through Lanes to
generate ATYPE by FTYPE by Lanes total X.
            if (_cntby[_aiter+_fiter+_liter2]>0)
              _links=_volvmtval[_aiter+_fiter+_liter2]/_cntvmtval[_aiter+_fiter+_liter2]
            else
              _links=0
            endif
            print list="\\"," ",_links(10.2C)," ", PRINTO=1
            _totvols=_totvols+_volvmtval[_aiter+_fiter+_liter2]
            _totcnts=_totcnts+_cntvmtval[_aiter+_fiter+_liter2]
            _supertotal=_supertotal+_volvmtval[_aiter+_fiter+_liter2]
            _supercnts=_supercnts+_cntvmtval[_aiter+_fiter+_liter2]
          ENDLOOP                                            ;^End Loop 6.

            if (_totcnts>0)
              _totvc=_totvols/_totcnts
            else
              _totvc=0
            endif
          print list="\\"," ",_totvc(10.2C), PRINTO=1
        endif                                                ;^End Condition 2.

      ENDLOOP                                                ;^End Loop 4.

      Print list= "-------------------------------------------------------------------------------
-----------------------------------------------", PRINTO=1
      print list="Totals", PRINTO=1

      LOOP _liter3=1,9,1                                     ;^Begin Loop 7: Cycles through Lanes for
                                                             ; current ATYPE in Loop 2.
        _lntotals=0                                          ;^Initialize Lane total X.
        _lncnts=0

        LOOP _aiter2=_aiter,599999,100                       ;^Begin Loop 8: Cycles through FTYPE for
current ATYPE
          if (_aiter2>_aiter+9999) BREAK                     ; in Loop 2 to generate Lane total X.
          _lntotiter=_aiter2+_liter3
          _lntotals=_lntotals+_volvmtval[_lntotiter]
```

```
        _lncnts=_lncnts+_cntvmtval[_lntotiter]
      ENDLOOP                                                ;^End Loop 8

      if (_lncnts>0)
        _lnvc=_lntotals/_lncnts
      else
        _lnvc=0
      endif
      print list="\\"," ",_lnvc(10.2C)," ", PRINTO=1
    ENDLOOP                                                  ;^End Loop 7

    if (_supercnts>0)
      _supervc=_supertotal/_supercnts
    else
      _supervc=0
    endif
    print list="\\"," ",_supervc(10.2C), PRINTO=1
    print list=" ", PRINTO=1
  endif                                                      ;^End Condition 1.

ENDLOOP                                                      ;^End Loop 2.

print list=" ", PRINTO=1
ENDLOOP                                                      ;^End Loop 1.

;-----------------2-DIGIT FACILITY TYPES BY TOTAL AREA TYPES--------------------

Print list= "Total Area Types ", PRINTO=1                    ;^Header
Print list= "                                                          Number of Lanes per Direction
", PRINTO=1
Print list= "FType         1           2           3           4           5           6           7
8           9       Totals", PRINTO=1
Print list= "-------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1

LOOP _fiter2=100,9900,100                                   ;^Begin Loop 9: Cycles through FTYPES to
get
  _fft2=int(_fiter2/100)                                     ; two-digit FTYPE.

  _tafvcheck=0                                               ;^Initialize FTYPE X checking variable.
  _tafcnts=0

  LOOP _liter5=1,9,1                                           ;^Begin Loop 10: Cycles through Lanes
for current
                                                             ; FTYPE in Loop 9.
    LOOP _aiter4= 100000,599999,10000                          ;^Begin Loop 11: Cycles through ATYPE
for
      _tafvcheck=_tafvcheck+_volvmtval[_aiter4+_fiter2+_liter5] ; current Lanes and FTYPE in order
to total X checking variable.
      _tafcnts=_tafcnts+_cntvmtval[_aiter4+_fiter2+_liter5]
    ENDLOOP                                                    ;^End Loop 11.

  ENDLOOP                                                    ;^End Loop 10.

  if (_tafvcheck>0)                                          ;^Begin Condition 3: If current FTYPE in
Loop 9
    print list= _fft2(2.0),"    ", PRINTO=1                 ; has X>0 continue to report X. Else skip
FTYPE.

    LOOP _liter4= 1,9,1                                        ;^Begin Loop 12: Cycles through Lanes
for current FTYPE
                                                             ; in Loop 9.
      _totftat=0                                            ;^Initialize FTYPE total X for all ATYPE.
      _totcnts=0

    LOOP _aiter3= 100000,599999,10000                          ;^Begin Loop 13: Cycles through ATYPE
for current Lanes in Loop 12
      _totftat=_totftat+_volvmtval[_aiter3+_fiter2+_liter4] ; in order to generate total X for
FTYPE by Lane for all ATYPE.
      _totcnts=_totcnts+_cntvmtval[_aiter3+_fiter2+_liter4]
    ENDLOOP                                                  ;^End Loop 13.
```

```
      if (_totcnts>0)
        _totvc=_totftat/_totcnts
      else
        _totvc=0
      endif
      print list="\\"," ",_totvc(10.2C)," ", PRINTO=1
    ENDLOOP                                                    ;^End Loop 12.
    if (_tafcnts>0)
      _tafvc=_tafvcheck/_tafcnts
    else
      _tafvc=0
    endif
    print list="\\"," ",_tafvc(10.2C), PRINTO=1
  endif                                                        ;^End Condition 3.

ENDLOOP                                                        ;^End Loop 9.

Print list= "-------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1
print list="Totals", PRINTO=1

_supertotal=0                                                  ;^Initialize all ATYPE total X.
_supercnts=0

LOOP _liter6=1,9,1                                             ;^Begin Loop 14: Cycles through Lanes.

  _lntotals=0                                                  ;^Initialize total X for Lanes.
  _lncnts=0

  LOOP _aiter5=100000,599999,100                                 ;^Begin Loop 15: Cycles through ATYPE
and
      _lntotiter=_aiter5+_liter6                               ; FTYPE in order to generate total X for
      _lntotals=_lntotals+_volvmtval[_lntotiter]                   ; Lanes.
      _lncnts=_lncnts+_cntvmtval[_lntotiter]
  ENDLOOP                                                      ;^End Loop 15.
  if (_lncnts>0)
    _lnvc=_lntotals/_lncnts
  else
    _lnvc=0
  endif
  print list="\\"," ",_lnvc(10.2C)," ", PRINTO=1
  _supertotal=_supertotal+_lntotals                            ;^Generate total X for all ATYPE.
  _supercnts=_supercnts+_lncnts
ENDLOOP                                                        ;^End Loop 14.
if (_supercnts>0)
  _supervc=_supertotal/_supercnts
else
  _supervc=0
endif
print list="\\"," ",_supervc(10.2C), PRINTO=1
print list=" ","\n ", PRINTO=1

;---------------1-DIGIT FACILITY TYPES BY 1-DIGIT AREA TYPES SUMMARY----------------

Print list= "Total Summary Area Types by Facility Types ", PRINTO=1 ;^Header
Print list= "                                                        Single Digit Facility Types
", PRINTO=1
Print list= "AType         1x          2x          3x          4x          5x          6x          7x
8x          9x      Totals", PRINTO=1
Print list= "-------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1

LOOP _a1iter2=100000,599999,100000                             ;^Begin Loop 16: Cycles through ATYPE by
10 to
  _aat1=int(_a1iter2/100000)                                   ; get single digit ATYPE.
  print list= _aat1(1.0),"x","    ", PRINTO=1

  _fttotal=0                                                   ;^Initialize total X for all ATYPE
  _ftcnts=0
```

```
    LOOP _f1iter=1000,9900,1000                              ;^Begin Loop 17: Cycles through FTYPE by
10 to
                                                             ; get single digit FTYPE.
      _totftlns=0                                              ;^Initialize total X for all FTYPE by
all Lanes.
      _totftcnts=0

      LOOP _fiter3=_f1iter,9900,100                          ;^Begin Loop 18: Cycles through two-digit
FTYPE
        if (_fiter3>_f1iter+999) BREAK                       ; for current single digit FTYPE in Loop
17.

        LOOP _aiter6=_a1iter2,599999,10000                   ;^Begin Loop 19: Cycles through two-digit
ATYPE
          if (_aiter6>_a1iter2+99999) BREAK                  ; for current single digit ATYPE in Loop
16.

            LOOP _liter7=1,9,1                                       ;^Begin Loop 20: Cycles through
Lanes for current FTYPE and ATYPE
              _totftlns=_totftlns+_volvmtval[_aiter6+_fiter3+_liter7]     ; in order to generate
total X for FTYPE by ATYPE.
              _totftcnts=_totftcnts+_cntvmtval[_aiter6+_fiter3+_liter7]
            ENDLOOP                                                   ;^End Loop 20.

        ENDLOOP                                              ;^End Loop 19.

      ENDLOOP                                                ;^End Loop 18.

      _fttotal=_fttotal+_totftlns                            ;^Generate total X for ATYPE.
      _ftcnts=_ftcnts+_totftcnts

      if (_totftcnts>0)
        _totftvc=_totftlns/_totftcnts
      else
        _totftvc=0
      endif
      print list="\\"," ",_totftvc(10.2C)," ", PRINTO=1
    ENDLOOP                                                  ;^End Loop 17.
    if (_ftcnts>0)
      _ftvc=_fttotal/_ftcnts
    else
      _ftvc=0
    endif
  print list="\\"," ",_ftvc(10.2c), PRINTO=1
ENDLOOP                                                      ;^End Loop 16.

Print list= "-------------------------------------------------------------------------------------
-----------------------------------------", PRINTO=1
print list="Totals", PRINTO=1

_supertotal=0                                                ;^Initialize overall total X.
_supercnts=0

LOOP _f1iter2=1000,9900,1000                                 ;^Begin Loop 21: Cycles through FTYPE by
10
                                                             ; to get single digit FTYPE.
  _ftotals=0                                                 ;^Initialize total X by FTYPE
  _ftcnts=0

  LOOP _fiter4=_f1iter2,9900,100                             ;^Begin Loop 22: Cycles through FTYPE by
1 to
    if (_fiter4>_f1iter2+999) BREAK                            ; get all two-digit FTYPE for current
FTYPE in
                                                             ; Loop 21.
    LOOP _liter8=1,9,1                                       ;^Begin Loop 23: Cycles through Lanes.

      LOOP _aiter7=100000,599999,10000                       ;^Begin Loop 24: Cycles through ATYPE in
order
        _ftotiter=_aiter7+_fiter4+_liter8                      ; to generate total X by single digit
FTYPE.
        _ftotals=_ftotals+_volvmtval[_ftotiter]
```

```
        _ftcnts=_ftcnts+_cntvmtval[_ftotiter]
      ENDLOOP                                                ;^End Loop 24.

    ENDLOOP                                                  ;^End Loop 23.

  ENDLOOP                                                    ;^End Loop 22.

  _supertotal=_supertotal+_ftotals                           ;^Generate overall total for all single
digit ATYPE
  _supercnts=_supercnts+_ftcnts                            ; by all single digit FTYPE.

  if (_ftcnts>0)
    _ftvc=_ftotals/_ftcnts
  else
    _ftvc=0
  endif
  print list="\\"," ",_ftvc(10.2C)," ", PRINTO=1
ENDLOOP                                                    ;^End Loop 21.
if (_supercnts>0)
  _supervc=_supertotal/_supercnts
else
  _supervc=0
endif
_vmtvolovercounts=_supervc
print list="\\"," ",_supervc(10.2C), PRINTO=1
print list=" ", PRINTO=1
;*************************************************************************
; END VMT VOLUME OVER COUNT REPORT
;*************************************************************************

;=============================================================================================
; BEGIN VHT VAL REPORT ------------ X = VHT on Links w/ Counts
;=============================================================================================
Print list=" ", PRINTO=1
Print
list="*************************************************************************************************
***********************************", PRINTO=1
Print                                                                                      list="*
*", PRINTO=1
Print list="*                                Vehicle Hours Traveled (VHT) using Volumes on Links
with Counts                                *", PRINTO=1
Print                                                                                      list="*
*", PRINTO=1
Print
list="*************************************************************************************************
***********************************", PRINTO=1
Print list=" ", PRINTO=1
;-----------------2-DIGIT FACILITY TYPES BY 2-DIGIT AREA TYPES--------------------

LOOP _a1iter=100000,599999,100000                          ;^Begin Loop 1: Cycles through Area Types
(ATYPE) by 10
  _aat1=int(_a1iter/100000)                                 ; in order to get single digit ATYPE.
  print list= "Area Type ",_aat1(1.0),"x Range:",
           "\n ", PRINTO=1

  LOOP _aiter=_a1iter,599999,10000                          ;^Begin Loop 2: Cycles through ATYPE by
1
    if (_aiter>_a1iter+99999) BREAK                         ; in order to get two-digit ATYPE.
    _aat2=int(_aiter/10000)

    _avcheck=0                                              ;^Initialize ATYPE X checking variable.

    LOOP _achkiter=_aiter,599999,1                          ;^Begin Loop 3: Cycles through Lanes and
Facility Types (FTYPE)
      if (_achkiter>_aiter+9999) BREAK                      ; for current ATYPE in Loop 2 and totals
X checking variable.
       _avcheck=_avcheck+_volvhtval[_achkiter]
    ENDLOOP                                                 ;^End Loop 3.

    if (_avcheck>0)                                         ;^Begin Condition 1: If current ATYPE in
Loop 2
```

```
                                                    ; has X>0 continue to report X. Else skip
ATYPE.
      _supertotal=0                                 ;^Initialize ATYPE total X.

      Print list= "Area Type ",_aat2(2.0), PRINTO=1 ;^Header
      Print list= "                                           Number of Lanes per Direction
", PRINTO=1
      Print list= "FType          1           2           3           4           5           6
7           8           9       Totals", PRINTO=1
      Print  list= "-------------------------------------------------------------------------------
-----------------------------------------------", PRINTO=1

      LOOP _fiter=100,9900,100                      ;^Begin Loop 4: Cycles through FTYPE
                                                    ; by 1 in order to get two-digit FTYPE.
       _vcheck=0                                    ;^Initialize FTYPE X checking variable.

       LOOP _liter=1,9,1                           ;^Begin Loop 5: Cycles through Lanes for
current
          _vcheck=_vcheck+_volvhtval[_aiter+_fiter+_liter]      ; FTYPE in Loop 4 and totals X
checking variable.
       ENDLOOP                                      ;^End Loop 5.

        if (_vcheck>0)                              ;^Begin Condition 2: If current FTYPE in
Loop 4
          _fft2=int(_fiter/100)                     ; has X>0 continue to report X. Else skip
FTYPE.
          print list= _fft2(2.0),"    ", PRINTO=1
          _totvols=0                                ;^Initialize FTYPE total X.

          LOOP _liter2=1,9,1                         ;^Begin Loop 6: Cycles through Lanes to
generate ATYPE by FTYPE by Lanes total X.
            print list="\\"," ",_volvhtval[_aiter+_fiter+_liter2](10.0C)," ", PRINTO=1
            _totvols=_totvols+_volvhtval[_aiter+_fiter+_liter2]
            _supertotal=_supertotal+_volvhtval[_aiter+_fiter+_liter2]
          ENDLOOP                                   ;^End Loop 6.

            print list="\\"," ",_totvols(10.0C), PRINTO=1
        endif                                       ;^End Condition 2.

      ENDLOOP                                       ;^End Loop 4.

      Print list= "-------------------------------------------------------------------------------
-----------------------------------------------", PRINTO=1
      print list="Totals", PRINTO=1

      LOOP _liter3=1,9,1                            ;^Begin Loop 7: Cycles through Lanes for
                                                    ; current ATYPE in Loop 2.
        _lntotals=0                                 ;^Initialize Lane total X.

        LOOP _aiter2=_aiter,599999,100              ;^Begin Loop 8: Cycles through FTYPE for
current ATYPE
          if (_aiter2>_aiter+9999) BREAK            ; in Loop 2 to generate Lane total X.
          _lntotiter=_aiter2+_liter3
          _lntotals=_lntotals+_volvhtval[_lntotiter]
        ENDLOOP                                     ;^End Loop 8

        print list="\\"," ",_lntotals(10.0C)," ", PRINTO=1
      ENDLOOP                                       ;^End Loop 7

      print list="\\"," ",_supertotal(10.0C), PRINTO=1
      print list=" ", PRINTO=1
    endif                                           ;^End Condition 1.

  ENDLOOP                                           ;^End Loop 2.

  print list=" ", PRINTO=1
ENDLOOP                                             ;^End Loop 1.

;-----------------2-DIGIT FACILITY TYPES BY TOTAL AREA TYPES--------------------

Print list= "Total Area Types ", PRINTO=1           ;^Header
```

```
Print list= "                                                                Number of Lanes per Direction
", PRINTO=1
Print list= "FType         1           2           3           4           5           6           7
8           9       Totals", PRINTO=1
Print list= "-------------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1

LOOP _fiter2=100,9900,100                                    ;^Begin Loop 9: Cycles through FTYPES to
get
  _fft2=int(_fiter2/100)                                     ; two-digit FTYPE.

  _tafvcheck=0                                               ;^Initialize FTYPE X checking variable.

  LOOP _liter5=1,9,1                                           ;^Begin Loop 10: Cycles through Lanes
for current
                                                             ; FTYPE in Loop 9.
    LOOP _aiter4= 100000,599999,10000                          ;^Begin Loop 11: Cycles through ATYPE
for
      _tafvcheck=_tafvcheck+_volvhtval[_aiter4+_fiter2+_liter5] ; current Lanes and FTYPE in order
to total X checking variable.
    ENDLOOP                                                    ;^End Loop 11.

  ENDLOOP                                                    ;^End Loop 10.

  if (_tafvcheck>0)                                          ;^Begin Condition 3: If current FTYPE in
Loop 9
    print list= _fft2(2.0),"    ", PRINTO=1                  ; has X>0 continue to report X. Else skip
FTYPE.

    LOOP _liter4= 1,9,1                                        ;^Begin Loop 12: Cycles through Lanes
for current FTYPE
                                                             ; in Loop 9.
        _totftat=0                                           ;^Initialize FTYPE total X for all ATYPE.

      LOOP _aiter3= 100000,599999,10000                        ;^Begin Loop 13: Cycles through ATYPE
for current Lanes in Loop 12
        _totftat=_totftat+_volvhtval[_aiter3+_fiter2+_liter4] ; in order to generate total X for
FTYPE by Lane for all ATYPE.
      ENDLOOP                                                ;^End Loop 13.

      print list="\\"," ",_totftat(10.0C)," ", PRINTO=1
    ENDLOOP                                                  ;^End Loop 12.

    print list="\\"," ",_tafvcheck(10.0C), PRINTO=1
  endif                                                      ;^End Condition 3.

ENDLOOP                                                      ;^End Loop 9.

Print list= "-------------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1
print list="Totals", PRINTO=1

_supertotal=0                                                ;^Initialize all ATYPE total X.

LOOP _liter6=1,9,1                                           ;^Begin Loop 14: Cycles through Lanes.

  _lntotals=0                                                ;^Initialize total X for Lanes.

  LOOP _aiter5=100000,599999,100                               ;^Begin Loop 15: Cycles through ATYPE
and
      _lntotiter=_aiter5+_liter6                             ; FTYPE in order to generate total X for
      _lntotals=_lntotals+_volvhtval[_lntotiter]               ; Lanes.
  ENDLOOP                                                    ;^End Loop 15.

  print list="\\"," ",_lntotals(10.0C)," ", PRINTO=1
  _supertotal=_supertotal+_lntotals                          ;^Generate total X for all ATYPE.

ENDLOOP                                                      ;^End Loop 14.
print list="\\"," ",_supertotal(10.0C), PRINTO=1
print list=" ","\n ", PRINTO=1
```

```
;---------------1-DIGIT FACILITY TYPES BY 1-DIGIT AREA TYPES SUMMARY----------------

Print list= "Total Summary Area Types by Facility Types ", PRINTO=1 ;^Header
Print list= "                                                    Single Digit Facility Types
", PRINTO=1
Print list= "AType         1x          2x          3x          4x          5x          6x          7x
8x          9x      Totals", PRINTO=1
Print list= "-------------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1

LOOP _a1iter2=100000,599999,100000                          ;^Begin Loop 16: Cycles through ATYPE by
10 to
  _aat1=int(_a1iter2/100000)                                ; get single digit ATYPE.
  print list= _aat1(1.0),"x","    ", PRINTO=1

  _fttotal=0                                                ;^Initialize total X for all ATYPE

    LOOP _f1iter=1000,9900,1000                             ;^Begin Loop 17: Cycles through FTYPE by
10 to
                                                            ; get single digit FTYPE.
      _totftlns=0                                             ;^Initialize total X for all FTYPE by
all Lanes.

      LOOP _fiter3=_f1iter,9900,100                         ;^Begin Loop 18: Cycles through two-digit
FTYPE
        if (_fiter3>_f1iter+999) BREAK                      ; for current single digit FTYPE in Loop
17.

        LOOP _aiter6=_a1iter2,599999,10000                  ;^Begin Loop 19: Cycles through two-digit
ATYPE
          if (_aiter6>_a1iter2+99999) BREAK                 ; for current single digit ATYPE in Loop
16.

            LOOP _liter7=1,9,1                                      ;^Begin Loop 20: Cycles through
Lanes for current FTYPE and ATYPE
              _totftlns=_totftlns+_volvhtval[_aiter6+_fiter3+_liter7]     ; in order to generate
total X for FTYPE by ATYPE.
            ENDLOOP                                                   ;^End Loop 20.

        ENDLOOP                                             ;^End Loop 19.

      ENDLOOP                                               ;^End Loop 18.

      _fttotal=_fttotal+_totftlns                           ;^Generate total X for ATYPE.

      print list="\\"," ",_totftlns(10.0C)," ", PRINTO=1
    ENDLOOP                                                 ;^End Loop 17.

  print list="\\"," ",_fttotal(10.0c), PRINTO=1
ENDLOOP                                                     ;^End Loop 16.

Print list= "-------------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1
print list="Totals", PRINTO=1

_supertotal=0                                               ;^Initialize overall total X.

LOOP _f1iter2=1000,9900,1000                                ;^Begin Loop 21: Cycles through FTYPE by
10
                                                            ; to get single digit FTYPE.
  _ftotals=0                                                ;^Initialize total X by FTYPE

  LOOP _fiter4=_f1iter2,9900,100                            ;^Begin Loop 22: Cycles through FTYPE by
1 to
    if (_fiter4>_f1iter2+999) BREAK                           ; get all two-digit FTYPE for current
FTYPE in
                                                            ; Loop 21.
    LOOP _liter8=1,9,1                                      ;^Begin Loop 23: Cycles through Lanes.

      LOOP _aiter7=100000,599999,10000                      ;^Begin Loop 24: Cycles through ATYPE in
order
```

```
        _ftotiter=_aiter7+_fiter4+_liter8                    ; to generate total X by single digit
FTYPE.
        _ftotals=_ftotals+_volvhtval[_ftotiter]
      ENDLOOP                                               ;^End Loop 24.

    ENDLOOP                                                 ;^End Loop 23.

  ENDLOOP                                                   ;^End Loop 22.

  _supertotal=_supertotal+_ftotals                          ;^Generate overall total for all single
digit ATYPE
                                                            ; by all single digit FTYPE.
  print list="\\"," ",_ftotals(10.0C)," ", PRINTO=1
ENDLOOP                                                     ;^End Loop 21.
_vhtvoloncounts=_supertotal
print list="\\"," ",_supertotal(10.0C), PRINTO=1
print list=" ", PRINTO=1
;*************************************************************************
; END VHT VAL REPORT
;*************************************************************************

;=============================================================================================
; BEGIN VHT Count REPORT ------------ X = Count VHT on Links w/ Counts
;=============================================================================================
Print list=" ", PRINTO=1
Print
list="*************************************************************************************************
***********************************", PRINTO=1
Print                                                                                       list="*
*", PRINTO=1
Print list="*                                Vehicle Hours Traveled (VHT) using Counts on Links with
Counts                                     *", PRINTO=1
Print                                                                                       list="*
*", PRINTO=1
Print
list="*************************************************************************************************
***********************************", PRINTO=1
Print list=" ", PRINTO=1
;-----------------2-DIGIT FACILITY TYPES BY 2-DIGIT AREA TYPES--------------------

LOOP _a1iter=100000,599999,100000                          ;^Begin Loop 1: Cycles through Area Types
(ATYPE) by 10
  _aat1=int(_a1iter/100000)                                 ; in order to get single digit ATYPE.
  print list= "Area Type ",_aat1(1.0),"x Range:",
           "\n ", PRINTO=1

  LOOP _aiter=_a1iter,599999,10000                          ;^Begin Loop 2: Cycles through ATYPE by
1
    if (_aiter>_a1iter+99999) BREAK                         ; in order to get two-digit ATYPE.
    _aat2=int(_aiter/10000)

    _avcheck=0                                              ;^Initialize ATYPE X checking variable.

    LOOP _achkiter=_aiter,599999,1                         ;^Begin Loop 3: Cycles through Lanes and
Facility Types (FTYPE)
      if (_achkiter>_aiter+9999) BREAK                      ; for current ATYPE in Loop 2 and totals
X checking variable.
      _avcheck=_avcheck+_cntvhtval[_achkiter]
    ENDLOOP                                                 ;^End Loop 3.

    if (_avcheck>0)                                        ;^Begin Condition 1: If current ATYPE in
Loop 2
                                                            ; has X>0 continue to report X. Else skip
ATYPE.
      _supertotal=0                                         ;^Initialize ATYPE total X.

      Print list= "Area Type ",_aat2(2.0), PRINTO=1         ;^Header
      Print list= "                                                 Number of Lanes per Direction
", PRINTO=1
      Print list= "FType          1           2           3           4           5           6
7           8           9       Totals", PRINTO=1
```

```
      Print list= "-------------------------------------------------------------------------------
-----------------------------------------------", PRINTO=1

      LOOP _fiter=100,9900,100                                ;^Begin Loop 4: Cycles through FTYPE
                                                              ; by 1 in order to get two-digit FTYPE.
       _vcheck=0                                              ;^Initialize FTYPE X checking variable.

       LOOP _liter=1,9,1                                     ;^Begin Loop 5: Cycles through Lanes for
current
         _vcheck=_vcheck+_cntvhtval[_aiter+_fiter+_liter]      ;  FTYPE  in  Loop  4  and  totals  X
checking variable.
       ENDLOOP                                                ;^End Loop 5.

        if (_vcheck>0)                                       ;^Begin Condition 2: If current FTYPE in
Loop 4
          _fft2=int(_fiter/100)                              ; has X>0 continue to report X. Else skip
FTYPE.
          print list= _fft2(2.0),"    ", PRINTO=1
          _totvols=0                                         ;^Initialize FTYPE total X.

          LOOP _liter2=1,9,1                                  ;^Begin Loop 6: Cycles through Lanes to
generate ATYPE by FTYPE by Lanes total X.
            print list="\\"," ",_cntvhtval[_aiter+_fiter+_liter2](10.0C)," ", PRINTO=1
            _totvols=_totvols+_cntvhtval[_aiter+_fiter+_liter2]
            _supertotal=_supertotal+_cntvhtval[_aiter+_fiter+_liter2]
          ENDLOOP                                            ;^End Loop 6.

            print list="\\"," ",_totvols(10.0C), PRINTO=1
        endif                                                ;^End Condition 2.

      ENDLOOP                                                ;^End Loop 4.

      Print list= "-------------------------------------------------------------------------------
-----------------------------------------------", PRINTO=1
      print list="Totals", PRINTO=1

      LOOP _liter3=1,9,1                                    ;^Begin Loop 7: Cycles through Lanes for
                                                            ; current ATYPE in Loop 2.
        _lntotals=0                                         ;^Initialize Lane total X.

        LOOP _aiter2=_aiter,599999,100                      ;^Begin Loop 8: Cycles through FTYPE for
current ATYPE
          if (_aiter2>_aiter+9999) BREAK                    ; in Loop 2 to generate Lane total X.
          _lntotiter=_aiter2+_liter3
          _lntotals=_lntotals+_cntvhtval[_lntotiter]
        ENDLOOP                                             ;^End Loop 8

        print list="\\"," ",_lntotals(10.0C)," ", PRINTO=1
      ENDLOOP                                               ;^End Loop 7


      print list="\\"," ",_supertotal(10.0C), PRINTO=1
      print list=" ", PRINTO=1
    endif                                                   ;^End Condition 1.

  ENDLOOP                                                   ;^End Loop 2.

  print list=" ", PRINTO=1
ENDLOOP                                                     ;^End Loop 1.

;-----------------2-DIGIT FACILITY TYPES BY TOTAL AREA TYPES--------------------

Print list= "Total Area Types ", PRINTO=1                   ;^Header
Print list= "                                                        Number of Lanes per Direction
", PRINTO=1
Print list= "FType         1           2           3           4           5           6           7
8           9       Totals", PRINTO=1
Print list= "-------------------------------------------------------------------------------------
-----------------------------------------", PRINTO=1
```

```
LOOP _fiter2=100,9900,100                                    ;^Begin Loop 9: Cycles through FTYPES to
get
  _fft2=int(_fiter2/100)                                     ; two-digit FTYPE.

  _tafvcheck=0                                               ;^Initialize FTYPE X checking variable.

  LOOP _liter5=1,9,1                                           ;^Begin Loop 10: Cycles through Lanes
for current
                                                             ; FTYPE in Loop 9.
    LOOP _aiter4= 100000,599999,10000                          ;^Begin Loop 11: Cycles through ATYPE
for
      _tafvcheck=_tafvcheck+_cntvhtval[_aiter4+_fiter2+_liter5] ; current Lanes and FTYPE in order
to total X checking variable.
    ENDLOOP                                                    ;^End Loop 11.

  ENDLOOP                                                    ;^End Loop 10.

  if (_tafvcheck>0)                                          ;^Begin Condition 3: If current FTYPE in
Loop 9
    print list= _fft2(2.0),"    ", PRINTO=1                  ; has X>0 continue to report X. Else skip
FTYPE.

    LOOP _liter4= 1,9,1                                        ;^Begin Loop 12: Cycles through Lanes
for current FTYPE
                                                             ; in Loop 9.
        _totftat=0                                           ;^Initialize FTYPE total X for all ATYPE.

      LOOP _aiter3= 100000,599999,10000                        ;^Begin Loop 13: Cycles through ATYPE
for current Lanes in Loop 12
        _totftat=_totftat+_cntvhtval[_aiter3+_fiter2+_liter4] ; in order to generate total X for
FTYPE by Lane for all ATYPE.
      ENDLOOP                                                  ;^End Loop 13.

      print list="\\"," ",_totftat(10.0C)," ", PRINTO=1
    ENDLOOP                                                    ;^End Loop 12.

    print list="\\"," ",_tafvcheck(10.0C), PRINTO=1
  endif                                                      ;^End Condition 3.

ENDLOOP                                                      ;^End Loop 9.

Print list= "-------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1
print list="Totals", PRINTO=1

_supertotal=0                                                ;^Initialize all ATYPE total X.

LOOP _liter6=1,9,1                                           ;^Begin Loop 14: Cycles through Lanes.

  _lntotals=0                                                ;^Initialize total X for Lanes.

  LOOP _aiter5=100000,599999,100                               ;^Begin Loop 15: Cycles through ATYPE
and
      _lntotiter=_aiter5+_liter6                             ; FTYPE in order to generate total X for
      _lntotals=_lntotals+_cntvhtval[_lntotiter]               ; Lanes.
  ENDLOOP                                                    ;^End Loop 15.

  print list="\\"," ",_lntotals(10.0C)," ", PRINTO=1
  _supertotal=_supertotal+_lntotals                          ;^Generate total X for all ATYPE.

ENDLOOP                                                      ;^End Loop 14.
print list="\\"," ",_supertotal(10.0C), PRINTO=1
print list=" ","\n ", PRINTO=1

;---------------1-DIGIT FACILITY TYPES BY 1-DIGIT AREA TYPES SUMMARY----------------

Print list= "Total Summary Area Types by Facility Types ", PRINTO=1 ;^Header
Print list= "                                                        Single Digit Facility Types
", PRINTO=1
Print list= "AType         1x          2x          3x          4x          5x          6x          7x
8x          9x      Totals", PRINTO=1
```

```
Print list= "-------------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1

LOOP _a1iter2=100000,599999,100000                          ;^Begin Loop 16: Cycles through ATYPE by
10 to
  _aat1=int(_a1iter2/100000)                                 ; get single digit ATYPE.
  print list= _aat1(1.0),"x","    ", PRINTO=1

  _fttotal=0                                                 ;^Initialize total X for all ATYPE

    LOOP _f1iter=1000,9900,1000                              ;^Begin Loop 17: Cycles through FTYPE by
10 to
                                                             ; get single digit FTYPE.
      _totftlns=0                                              ;^Initialize total X for all FTYPE by
all Lanes.

      LOOP _fiter3=_f1iter,9900,100                         ;^Begin Loop 18: Cycles through two-digit
FTYPE
        if (_fiter3>_f1iter+999) BREAK                       ; for current single digit FTYPE in Loop
17.

        LOOP _aiter6=_a1iter2,599999,10000                  ;^Begin Loop 19: Cycles through two-digit
ATYPE
          if (_aiter6>_a1iter2+99999) BREAK                  ; for current single digit ATYPE in Loop
16.

            LOOP _liter7=1,9,1                                          ;^Begin Loop 20: Cycles through
Lanes for current FTYPE and ATYPE
              _totftlns=_totftlns+_cntvhtval[_aiter6+_fiter3+_liter7]    ; in order to generate
total X for FTYPE by ATYPE.
            ENDLOOP                                                      ;^End Loop 20.

        ENDLOOP                                              ;^End Loop 19.

      ENDLOOP                                                ;^End Loop 18.

      _fttotal=_fttotal+_totftlns                            ;^Generate total X for ATYPE.

      print list="\\"," ",_totftlns(10.0C)," ", PRINTO=1
    ENDLOOP                                                  ;^End Loop 17.

  print list="\\"," ",_fttotal(10.0c), PRINTO=1
ENDLOOP                                                      ;^End Loop 16.

Print list= "-------------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1
print list="Totals", PRINTO=1

_supertotal=0                                                ;^Initialize overall total X.

LOOP _f1iter2=1000,9900,1000                                 ;^Begin Loop 21: Cycles through FTYPE by
10
                                                             ; to get single digit FTYPE.
  _ftotals=0                                                 ;^Initialize total X by FTYPE

  LOOP _fiter4=_f1iter2,9900,100                             ;^Begin Loop 22: Cycles through FTYPE by
1 to
    if (_fiter4>_f1iter2+999) BREAK                            ; get all two-digit FTYPE for current
FTYPE in
                                                             ; Loop 21.
    LOOP _liter8=1,9,1                                       ;^Begin Loop 23: Cycles through Lanes.

      LOOP _aiter7=100000,599999,10000                       ;^Begin Loop 24: Cycles through ATYPE in
order
        _ftotiter=_aiter7+_fiter4+_liter8                      ; to generate total X by single digit
FTYPE.
        _ftotals=_ftotals+_cntvhtval[_ftotiter]
      ENDLOOP                                                ;^End Loop 24.

    ENDLOOP                                                  ;^End Loop 23.
```

```
  ENDLOOP                                                    ;^End Loop 22.

  _supertotal=_supertotal+_ftotals                           ;^Generate overall total for all single
digit ATYPE
                                                             ; by all single digit FTYPE.
  print list="\\"," ",_ftotals(10.0C)," ", PRINTO=1
ENDLOOP                                                      ;^End Loop 21.
_vhtcountsoncounts=_supertotal
print list="\\"," ",_supertotal(10.0C), PRINTO=1
print list=" ", PRINTO=1
;*********************************************************************
; END COUNT VHT REPORT
;*********************************************************************

;=============================================================================================
; BEGIN VOLUME/COUNT VHT REPORT ------------- X = Volumes over Counts VHT on Links w/ Counts
;=============================================================================================
Print list=" ", PRINTO=1
Print
list="*********************************************************************************************
***********************************", PRINTO=1
Print                                                                                    list="*
*", PRINTO=1
Print list="*                                  VHT Volume over Count Ratios on Links with Counts
*", PRINTO=1
Print                                                                                    list="*
*", PRINTO=1
Print
list="*********************************************************************************************
***********************************", PRINTO=1
Print list=" ", PRINTO=1
;-----------------2-DIGIT FACILITY TYPES BY 2-DIGIT AREA TYPES--------------------

LOOP _a1iter=100000,599999,100000                           ;^Begin Loop 1: Cycles through Area Types
(ATYPE) by 10
  _aat1=int(_a1iter/100000)                                  ; in order to get single digit ATYPE.
  print list= "Area Type ",_aat1(1.0),"x Range:",
           "\n ", PRINTO=1

  LOOP _aiter=_a1iter,599999,10000                           ;^Begin Loop 2: Cycles through ATYPE by
1
    if (_aiter>_a1iter+99999) BREAK                          ; in order to get two-digit ATYPE.
    _aat2=int(_aiter/10000)

    _avcheck=0                                               ;^Initialize ATYPE X checking variable.

    LOOP _achkiter=_aiter,599999,1                           ;^Begin Loop 3: Cycles through Lanes and
Facility Types (FTYPE)
      if (_achkiter>_aiter+9999) BREAK                       ; for current ATYPE in Loop 2 and totals
X checking variable.
      _avcheck=_avcheck+_vcntby[_achkiter]
    ENDLOOP                                                  ;^End Loop 3.

    if (_avcheck>0)                                          ;^Begin Condition 1: If current ATYPE in
Loop 2
                                                             ; has X>0 continue to report X. Else skip
ATYPE.
      _supertotal=0                                          ;^Initialize ATYPE total X.
      _supercnts=0

      Print list= "Area Type ",_aat2(2.0), PRINTO=1          ;^Header
      Print list= "                                                    Number of Lanes per Direction
", PRINTO=1
      Print list= "FType          1           2           3           4           5           6
7           8           9       Totals", PRINTO=1
      Print list= "-------------------------------------------------------------------------------
-----------------------------------------------", PRINTO=1

      LOOP _fiter=100,9900,100                               ;^Begin Loop 4: Cycles through FTYPE
                                                             ; by 1 in order to get two-digit FTYPE.
       _vcheck=0                                             ;^Initialize FTYPE X checking variable.
```

```
       LOOP _liter=1,9,1                                    ;^Begin Loop 5: Cycles through Lanes for
current
          _vcheck=_vcheck+_vcntby[_aiter+_fiter+_liter]     ; FTYPE in Loop 4 and totals X checking
variable.
       ENDLOOP                                              ;^End Loop 5.

        if (_vcheck>0)                                      ;^Begin Condition 2: If current FTYPE in
Loop 4
          _fft2=int(_fiter/100)                             ; has X>0 continue to report X. Else skip
FTYPE.
          print list= _fft2(2.0),"    ", PRINTO=1
          _totvols=0                                        ;^Initialize FTYPE total X.
          _totcnts=0

           LOOP _liter2=1,9,1                               ;^Begin Loop 6: Cycles through Lanes to
generate ATYPE by FTYPE by Lanes total X.
            if (_cntby[_aiter+_fiter+_liter2]>0)
              _links=_volvhtval[_aiter+_fiter+_liter2]/_cntvhtval[_aiter+_fiter+_liter2]
            else
              _links=0
            endif
            print list="\\"," ",_links(10.2C)," ", PRINTO=1
            _totvols=_totvols+_volvhtval[_aiter+_fiter+_liter2]
            _totcnts=_totcnts+_cntvhtval[_aiter+_fiter+_liter2]
            _supertotal=_supertotal+_volvhtval[_aiter+_fiter+_liter2]
            _supercnts=_supercnts+_cntvhtval[_aiter+_fiter+_liter2]
          ENDLOOP                                           ;^End Loop 6.

            if (_totcnts>0)
              _totvc=_totvols/_totcnts
            else
              _totvc=0
            endif
          print list="\\"," ",_totvc(10.2C), PRINTO=1
        endif                                               ;^End Condition 2.

      ENDLOOP                                               ;^End Loop 4.

      Print list= "-------------------------------------------------------------------------------
-------------------------------------------------", PRINTO=1
      print list="Totals", PRINTO=1

      LOOP _liter3=1,9,1                                    ;^Begin Loop 7: Cycles through Lanes for
                                                            ; current ATYPE in Loop 2.
        _lntotals=0                                         ;^Initialize Lane total X.
        _lncnts=0

        LOOP _aiter2=_aiter,599999,100                      ;^Begin Loop 8: Cycles through FTYPE for
current ATYPE
          if (_aiter2>_aiter+9999) BREAK                    ; in Loop 2 to generate Lane total X.
          _lntotiter=_aiter2+_liter3
          _lntotals=_lntotals+_volvhtval[_lntotiter]
          _lncnts=_lncnts+_cntvhtval[_lntotiter]
        ENDLOOP                                             ;^End Loop 8

        if (_lncnts>0)
          _lnvc=_lntotals/_lncnts
        else
          _lnvc=0
        endif
        print list="\\"," ",_lnvc(10.2C)," ", PRINTO=1
      ENDLOOP                                               ;^End Loop 7

      if (_supercnts>0)
        _supervc=_supertotal/_supercnts
      else
        _supervc=0
      endif
      print list="\\"," ",_supervc(10.2C), PRINTO=1
      print list=" ", PRINTO=1
```

```
    endif                                                    ;^End Condition 1.

  ENDLOOP                                                    ;^End Loop 2.

  print list=" ", PRINTO=1
ENDLOOP                                                      ;^End Loop 1.

;-----------------2-DIGIT FACILITY TYPES BY TOTAL AREA TYPES--------------------

Print list= "Total Area Types ", PRINTO=1                    ;^Header
Print list= "                                                        Number of Lanes per Direction
", PRINTO=1
Print list= "FType         1           2           3           4           5           6           7
8           9       Totals", PRINTO=1
Print list= "-------------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1

LOOP _fiter2=100,9900,100                                   ;^Begin Loop 9: Cycles through FTYPES to
get
  _fft2=int(_fiter2/100)                                     ; two-digit FTYPE.

  _tafvcheck=0                                               ;^Initialize FTYPE X checking variable.
  _tafcnts=0

  LOOP _liter5=1,9,1                                           ;^Begin Loop 10: Cycles through Lanes
for current
                                                             ; FTYPE in Loop 9.
    LOOP _aiter4= 100000,599999,10000                          ;^Begin Loop 11: Cycles through ATYPE
for
      _tafvcheck=_tafvcheck+_volvhtval[_aiter4+_fiter2+_liter5] ; current Lanes and FTYPE in order
to total X checking variable.
      _tafcnts=_tafcnts+_cntvhtval[_aiter4+_fiter2+_liter5]
    ENDLOOP                                                    ;^End Loop 11.

  ENDLOOP                                                    ;^End Loop 10.

  if (_tafvcheck>0)                                          ;^Begin Condition 3: If current FTYPE in
Loop 9
    print list= _fft2(2.0),"    ", PRINTO=1                  ; has X>0 continue to report X. Else skip
FTYPE.

    LOOP _liter4= 1,9,1                                        ;^Begin Loop 12: Cycles through Lanes
for current FTYPE
                                                             ; in Loop 9.
        _totftat=0                                           ;^Initialize FTYPE total X for all ATYPE.
        _totcnts=0

      LOOP _aiter3= 100000,599999,10000                        ;^Begin Loop 13: Cycles through ATYPE
for current Lanes in Loop 12
        _totftat=_totftat+_volvhtval[_aiter3+_fiter2+_liter4] ; in order to generate total X for
FTYPE by Lane for all ATYPE.
        _totcnts=_totcnts+_cntvhtval[_aiter3+_fiter2+_liter4]
      ENDLOOP                                                ;^End Loop 13.
      if (_totcnts>0)
        _totvc=_totftat/_totcnts
      else
        _totvc=0
      endif
      print list="\\"," ",_totvc(10.2C)," ", PRINTO=1
    ENDLOOP                                                  ;^End Loop 12.
    if (_tafcnts>0)
      _tafvc=_tafvcheck/_tafcnts
    else
      _tafvc=0
    endif
    print list="\\"," ",_tafvc(10.2C), PRINTO=1
  endif                                                      ;^End Condition 3.

ENDLOOP                                                      ;^End Loop 9.
```

```
Print list= "-------------------------------------------------------------------------------------------
-----------------------------------------", PRINTO=1
print list="Totals", PRINTO=1

_supertotal=0                                          ;^Initialize all ATYPE total X.
_supercnts=0

LOOP _liter6=1,9,1                                     ;^Begin Loop 14: Cycles through Lanes.

  _lntotals=0                                          ;^Initialize total X for Lanes.
  _lncnts=0

  LOOP _aiter5=100000,599999,100                         ;^Begin Loop 15: Cycles through ATYPE
and
      _lntotiter=_aiter5+_liter6                       ; FTYPE in order to generate total X for
      _lntotals=_lntotals+_volvhtval[_lntotiter]           ; Lanes.
      _lncnts=_lncnts+_cntvhtval[_lntotiter]
  ENDLOOP                                              ;^End Loop 15.
  if (_lncnts>0)
    _lnvc=_lntotals/_lncnts
  else
    _lnvc=0
  endif
  print list="\\"," ",_lnvc(10.2C)," ", PRINTO=1
  _supertotal=_supertotal+_lntotals                    ;^Generate total X for all ATYPE.
  _supercnts=_supercnts+_lncnts
ENDLOOP                                                ;^End Loop 14.
if (_supercnts>0)
  _supervc=_supertotal/_supercnts
else
  _supervc=0
endif
print list="\\"," ",_supervc(10.2C), PRINTO=1
print list=" ","\n ", PRINTO=1

;---------------1-DIGIT FACILITY TYPES BY 1-DIGIT AREA TYPES SUMMARY----------------

Print list= "Total Summary Area Types by Facility Types ", PRINTO=1 ;^Header
Print list= "                                                    Single Digit Facility Types
", PRINTO=1
Print list= "AType         1x          2x          3x          4x          5x          6x          7x
8x          9x      Totals", PRINTO=1
Print list= "-------------------------------------------------------------------------------------------
-----------------------------------------", PRINTO=1

LOOP _a1iter2=100000,599999,100000                     ;^Begin Loop 16: Cycles through ATYPE by
10 to
  _aat1=int(_a1iter2/100000)                           ; get single digit ATYPE.
  print list= _aat1(1.0),"x","    ", PRINTO=1

  _fttotal=0                                           ;^Initialize total X for all ATYPE
  _ftcnts=0

    LOOP _f1iter=1000,9900,1000                        ;^Begin Loop 17: Cycles through FTYPE by
10 to
                                                       ; get single digit FTYPE.
      _totftlns=0                                        ;^Initialize total X for all FTYPE by
all Lanes.
      _totftcnts=0

      LOOP _fiter3=_f1iter,9900,100                    ;^Begin Loop 18: Cycles through two-digit
FTYPE
        if (_fiter3>_f1iter+999) BREAK                 ; for current single digit FTYPE in Loop
17.

        LOOP _aiter6=_a1iter2,599999,10000             ;^Begin Loop 19: Cycles through two-digit
ATYPE
          if (_aiter6>_a1iter2+99999) BREAK            ; for current single digit ATYPE in Loop
16.
```

```
            LOOP _liter7=1,9,1                                          ;^Begin Loop 20: Cycles through
Lanes for current FTYPE and ATYPE
              _totftlns=_totftlns+_volvhtval[_aiter6+_fiter3+_liter7]    ; in order to generate
total X for FTYPE by ATYPE.
              _totftcnts=_totftcnts+_cntvhtval[_aiter6+_fiter3+_liter7]
            ENDLOOP                                                      ;^End Loop 20.

        ENDLOOP                                                 ;^End Loop 19.

      ENDLOOP                                                   ;^End Loop 18.

      _fttotal=_fttotal+_totftlns                               ;^Generate total X for ATYPE.
      _ftcnts=_ftcnts+_totftcnts

      if (_totftcnts>0)
        _totftvc=_totftlns/_totftcnts
      else
        _totftvc=0
      endif
      print list="\\"," ",_totftvc(10.2C)," ", PRINTO=1
    ENDLOOP                                                     ;^End Loop 17.
    if (_ftcnts>0)
      _ftvc=_fttotal/_ftcnts
    else
      _ftvc=0
    endif
  print list="\\"," ",_ftvc(10.2c), PRINTO=1
ENDLOOP                                                         ;^End Loop 16.

Print list= "-------------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1
print list="Totals", PRINTO=1

_supertotal=0                                                   ;^Initialize overall total X.
_supercnts=0

LOOP _f1iter2=1000,9900,1000                                    ;^Begin Loop 21: Cycles through FTYPE by
10
                                                                ; to get single digit FTYPE.
  _ftotals=0                                                    ;^Initialize total X by FTYPE
  _ftcnts=0

  LOOP _fiter4=_f1iter2,9900,100                                ;^Begin Loop 22: Cycles through FTYPE by
1 to
    if (_fiter4>_f1iter2+999) BREAK                               ; get all two-digit FTYPE for current
FTYPE in
                                                                ; Loop 21.
    LOOP _liter8=1,9,1                                          ;^Begin Loop 23: Cycles through Lanes.

      LOOP _aiter7=100000,599999,10000                          ;^Begin Loop 24: Cycles through ATYPE in
order
        _ftotiter=_aiter7+_fiter4+_liter8                         ; to generate total X by single digit
FTYPE.
        _ftotals=_ftotals+_volvhtval[_ftotiter]
        _ftcnts=_ftcnts+_cntvhtval[_ftotiter]
      ENDLOOP                                                   ;^End Loop 24.

    ENDLOOP                                                     ;^End Loop 23.

  ENDLOOP                                                       ;^End Loop 22.

  _supertotal=_supertotal+_ftotals                               ;^Generate overall total for all single
digit ATYPE
  _supercnts=_supercnts+_ftcnts                               ; by all single digit FTYPE.

  if (_ftcnts>0)
    _ftvc=_ftotals/_ftcnts
  else
    _ftvc=0
  endif
```

```
  print list="\\"," ",_ftvc(10.2C)," ", PRINTO=1
ENDLOOP                                                    ;^End Loop 21.
if (_supercnts>0)
  _supervc=_supertotal/_supercnts
else
  _supervc=0
endif
_vhtvolovercounts=_supervc
print list="\\"," ",_supervc(10.2C), PRINTO=1
print list=" ", PRINTO=1
;*********************************************************************
; END VHT VOLUME OVER COUNT REPORT
;*********************************************************************

;=============================================================================================
; BEGIN VOLUME REPORT ------------- X = Volumes on Links w/ Counts
;=============================================================================================
Print list=" ", PRINTO=1
Print
list="*************************************************************************************
***********************************", PRINTO=1
Print                                                                                  list="*
*", PRINTO=1
Print list="*                                                Estimated Volumes on Links with Counts
*", PRINTO=1
Print                                                                                  list="*
*", PRINTO=1
Print
list="*************************************************************************************
***********************************", PRINTO=1
Print list=" ", PRINTO=1
;-----------------2-DIGIT FACILITY TYPES BY 2-DIGIT AREA TYPES--------------------

LOOP _a1iter=100000,599999,100000                         ;^Begin Loop 1: Cycles through Area Types
(ATYPE) by 10
  _aat1=int(_a1iter/100000)                                ; in order to get single digit ATYPE.
  print list= "Area Type ",_aat1(1.0),"x Range:",
           "\n ", PRINTO=1

  LOOP _aiter=_a1iter,599999,10000                         ;^Begin Loop 2: Cycles through ATYPE by
1
    if (_aiter>_a1iter+99999) BREAK                        ; in order to get two-digit ATYPE.
    _aat2=int(_aiter/10000)

    _avcheck=0                                             ;^Initialize ATYPE X checking variable.

    LOOP _achkiter=_aiter,599999,1                         ;^Begin Loop 3: Cycles through Lanes and
Facility Types (FTYPE)
      if (_achkiter>_aiter+9999) BREAK                     ; for current ATYPE in Loop 2 and totals
X checking variable.
      _avcheck=_avcheck+_volby[_achkiter]
    ENDLOOP                                                ;^End Loop 3.

    if (_avcheck>0)                                        ;^Begin Condition 1: If current ATYPE in
Loop 2
                                                           ; has X>0 continue to report X. Else skip
ATYPE.
      _supertotal=0                                        ;^Initialize ATYPE total X.

      Print list= "Area Type ",_aat2(2.0), PRINTO=1        ;^Header
      Print list= "                                                Number of Lanes per Direction
", PRINTO=1
      Print list= "FType          1           2           3           4           5           6
7           8           9       Totals", PRINTO=1
      Print list= "-------------------------------------------------------------------------------
-----------------------------------------------", PRINTO=1

      LOOP _fiter=100,9900,100                             ;^Begin Loop 4: Cycles through FTYPE
                                                           ; by 1 in order to get two-digit FTYPE.
       _vcheck=0                                           ;^Initialize FTYPE X checking variable.
```

```
       LOOP _liter=1,9,1                                    ;^Begin Loop 5: Cycles through Lanes for
current
          _vcheck=_vcheck+_volby[_aiter+_fiter+_liter]     ; FTYPE in Loop 4 and totals X checking
variable.
       ENDLOOP                                              ;^End Loop 5.

        if (_vcheck>0)                                      ;^Begin Condition 2: If current FTYPE in
Loop 4
          _fft2=int(_fiter/100)                            ; has X>0 continue to report X. Else skip
FTYPE.
          print list= _fft2(2.0),"    ", PRINTO=1
          _totvols=0                                        ;^Initialize FTYPE total X.

          LOOP _liter2=1,9,1                                ;^Begin Loop 6: Cycles through Lanes to
generate ATYPE by FTYPE by Lanes total X.
            print list="\\"," ",_volby[_aiter+_fiter+_liter2](10.0C)," ", PRINTO=1
            _totvols=_totvols+_volby[_aiter+_fiter+_liter2]
            _supertotal=_supertotal+_volby[_aiter+_fiter+_liter2]
          ENDLOOP                                           ;^End Loop 6.

            print list="\\"," ",_totvols(10.0C), PRINTO=1
        endif                                               ;^End Condition 2.

      ENDLOOP                                               ;^End Loop 4.

      Print list= "-------------------------------------------------------------------------------
-------------------------------------------------", PRINTO=1
      print list="Totals", PRINTO=1

      LOOP _liter3=1,9,1                                    ;^Begin Loop 7: Cycles through Lanes for
                                                            ; current ATYPE in Loop 2.
        _lntotals=0                                         ;^Initialize Lane total X.

        LOOP _aiter2=_aiter,599999,100                      ;^Begin Loop 8: Cycles through FTYPE for
current ATYPE
          if (_aiter2>_aiter+9999) BREAK                    ; in Loop 2 to generate Lane total X.
          _lntotiter=_aiter2+_liter3
          _lntotals=_lntotals+_volby[_lntotiter]
        ENDLOOP                                             ;^End Loop 8

        print list="\\"," ",_lntotals(10.0C)," ", PRINTO=1
      ENDLOOP                                               ;^End Loop 7

      print list="\\"," ",_supertotal(10.0C), PRINTO=1
      print list=" ", PRINTO=1
    endif                                                   ;^End Condition 1.

  ENDLOOP                                                   ;^End Loop 2.

  print list=" ", PRINTO=1
ENDLOOP                                                     ;^End Loop 1.

;-----------------2-DIGIT FACILITY TYPES BY TOTAL AREA TYPES--------------------

Print list= "Total Area Types ", PRINTO=1                   ;^Header
Print list= "                                                        Number of Lanes per Direction
", PRINTO=1
Print list= "FType         1          2          3          4          5          6          7
8          9       Totals", PRINTO=1
Print list= "-------------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1

LOOP _fiter2=100,9900,100                                   ;^Begin Loop 9: Cycles through FTYPES to
get
  _fft2=int(_fiter2/100)                                    ; two-digit FTYPE.

  _tafvcheck=0                                              ;^Initialize FTYPE X checking variable.

  LOOP _liter5=1,9,1                                          ;^Begin Loop 10: Cycles through Lanes
for current
                                                            ; FTYPE in Loop 9.
```

```
    LOOP _aiter4= 100000,599999,10000                          ;^Begin Loop 11: Cycles through ATYPE
for
      _tafvcheck=_tafvcheck+_volby[_aiter4+_fiter2+_liter5] ; current Lanes and FTYPE in order to
total X checking variable.
    ENDLOOP                                                    ;^End Loop 11.

  ENDLOOP                                                     ;^End Loop 10.

  if (_tafvcheck>0)                                           ;^Begin Condition 3: If current FTYPE in
Loop 9
    print list= _fft2(2.0),"    ", PRINTO=1                  ; has X>0 continue to report X. Else skip
FTYPE.

    LOOP _liter4= 1,9,1                                        ;^Begin Loop 12: Cycles through Lanes
for current FTYPE
                                                              ; in Loop 9.
        _totftat=0                                           ;^Initialize FTYPE total X for all ATYPE.

      LOOP _aiter3= 100000,599999,10000                        ;^Begin Loop 13: Cycles through ATYPE
for current Lanes in Loop 12
       _totftat=_totftat+_volby[_aiter3+_fiter2+_liter4] ; in order to generate total X for FTYPE
by Lane for all ATYPE.
      ENDLOOP                                                 ;^End Loop 13.

      print list="\\"," ",_totftat(10.0C)," ", PRINTO=1
    ENDLOOP                                                   ;^End Loop 12.

    print list="\\"," ",_tafvcheck(10.0C), PRINTO=1
  endif                                                       ;^End Condition 3.

ENDLOOP                                                       ;^End Loop 9.

Print list= "-------------------------------------------------------------------------------------
-----------------------------------------", PRINTO=1
print list="Totals", PRINTO=1

_supertotal=0                                                 ;^Initialize all ATYPE total X.

LOOP _liter6=1,9,1                                            ;^Begin Loop 14: Cycles through Lanes.

  _lntotals=0                                                 ;^Initialize total X for Lanes.

  LOOP _aiter5=100000,599999,100                               ;^Begin Loop 15: Cycles through ATYPE
and
      _lntotiter=_aiter5+_liter6                             ; FTYPE in order to generate total X for
      _lntotals=_lntotals+_volby[_lntotiter]                 ; Lanes.
  ENDLOOP                                                     ;^End Loop 15.

  print list="\\"," ",_lntotals(10.0C)," ", PRINTO=1
  _supertotal=_supertotal+_lntotals                           ;^Generate total X for all ATYPE.

ENDLOOP                                                       ;^End Loop 14.
print list="\\"," ",_supertotal(10.0C), PRINTO=1
print list=" ","\n ", PRINTO=1

;---------------1-DIGIT FACILITY TYPES BY 1-DIGIT AREA TYPES SUMMARY----------------

Print list= "Total Summary Area Types by Facility Types ", PRINTO=1 ;^Header
Print list= "                                                    Single Digit Facility Types
", PRINTO=1
Print list= "AType         1x          2x          3x          4x          5x          6x          7x
8x          9x      Totals", PRINTO=1
Print list= "-------------------------------------------------------------------------------------
-----------------------------------------", PRINTO=1

LOOP _a1iter2=100000,599999,100000                            ;^Begin Loop 16: Cycles through ATYPE by
10 to
  _aat1=int(_a1iter2/100000)                                  ; get single digit ATYPE.
  print list= _aat1(1.0),"x","    ", PRINTO=1

  _fttotal=0                                                  ;^Initialize total X for all ATYPE
```

```
    LOOP _f1iter=1000,9900,1000                                ;^Begin Loop 17: Cycles through FTYPE by
10 to
                                                               ; get single digit FTYPE.
      _totftlns=0                                                ;^Initialize total X for all FTYPE by
all Lanes.

      LOOP _fiter3=_f1iter,9900,100                            ;^Begin Loop 18: Cycles through two-digit
FTYPE
        if (_fiter3>_f1iter+999) BREAK                          ; for current single digit FTYPE in Loop
17.

        LOOP _aiter6=_a1iter2,599999,10000                     ;^Begin Loop 19: Cycles through two-digit
ATYPE
          if (_aiter6>_a1iter2+99999) BREAK                     ; for current single digit ATYPE in Loop
16.

            LOOP _liter7=1,9,1                                                ;^Begin Loop 20: Cycles through
Lanes for current FTYPE and ATYPE
              _totftlns=_totftlns+_volby[_aiter6+_fiter3+_liter7]   ; in order to generate total X
for FTYPE by ATYPE.
            ENDLOOP                                                            ;^End Loop 20.

        ENDLOOP                                                ;^End Loop 19.

      ENDLOOP                                                  ;^End Loop 18.

      _fttotal=_fttotal+_totftlns                              ;^Generate total X for ATYPE.

      print list="\\"," ",_totftlns(10.0C)," ", PRINTO=1
    ENDLOOP                                                    ;^End Loop 17.

  print list="\\"," ",_fttotal(10.0c), PRINTO=1
ENDLOOP                                                        ;^End Loop 16.

Print list= "---------------------------------------------------------------------------------------
-----------------------------------------", PRINTO=1
print list="Totals", PRINTO=1

_supertotal=0                                                  ;^Initialize overall total X.

LOOP _f1iter2=1000,9900,1000                                   ;^Begin Loop 21: Cycles through FTYPE by
10
                                                               ; to get single digit FTYPE.
  _ftotals=0                                                   ;^Initialize total X by FTYPE

  LOOP _fiter4=_f1iter2,9900,100                               ;^Begin Loop 22: Cycles through FTYPE by
1 to
    if (_fiter4>_f1iter2+999) BREAK                              ; get all two-digit FTYPE for current
FTYPE in
                                                               ; Loop 21.
    LOOP _liter8=1,9,1                                         ;^Begin Loop 23: Cycles through Lanes.

      LOOP _aiter7=100000,599999,10000                         ;^Begin Loop 24: Cycles through ATYPE in
order
        _ftotiter=_aiter7+_fiter4+_liter8                        ; to generate total X by single digit
FTYPE.
        _ftotals=_ftotals+_volby[_ftotiter]
      ENDLOOP                                                  ;^End Loop 24.

    ENDLOOP                                                    ;^End Loop 23.

  ENDLOOP                                                      ;^End Loop 22.

  _supertotal=_supertotal+_ftotals                              ;^Generate overall total for all single
digit ATYPE
                                                               ; by all single digit FTYPE.
  print list="\\"," ",_ftotals(10.0C)," ", PRINTO=1
ENDLOOP                                                        ;^End Loop 21.

print list="\\"," ",_supertotal(10.0C), PRINTO=1
```

```
print list=" ", PRINTO=1
;*********************************************************************
; END VOLUME REPORT
;*********************************************************************

;=============================================================================================
; BEGIN Count REPORT ------------- X = Count on Links w/ Counts
;=============================================================================================
Print list=" ", PRINTO=1
Print
list="*************************************************************************************************
***********************************", PRINTO=1
Print                                                                                          list="*
*", PRINTO=1
Print list="*                                                 Observed Counts on Links with Counts
*", PRINTO=1
Print                                                                                          list="*
*", PRINTO=1
Print
list="*************************************************************************************************
***********************************", PRINTO=1
Print list=" ", PRINTO=1
;-----------------2-DIGIT FACILITY TYPES BY 2-DIGIT AREA TYPES--------------------

LOOP _a1iter=100000,599999,100000                          ;^Begin Loop 1: Cycles through Area Types
(ATYPE) by 10
  _aat1=int(_a1iter/100000)                                 ; in order to get single digit ATYPE.
  print list= "Area Type ",_aat1(1.0),"x Range:",
           "\n ", PRINTO=1

  LOOP _aiter=_a1iter,599999,10000                          ;^Begin Loop 2: Cycles through ATYPE by
1
    if (_aiter>_a1iter+99999) BREAK                         ; in order to get two-digit ATYPE.
    _aat2=int(_aiter/10000)

    _avcheck=0                                              ;^Initialize ATYPE X checking variable.

    LOOP _achkiter=_aiter,599999,1                          ;^Begin Loop 3: Cycles through Lanes and
Facility Types (FTYPE)
      if (_achkiter>_aiter+9999) BREAK                      ; for current ATYPE in Loop 2 and totals
X checking variable.
      _avcheck=_avcheck+_cntby[_achkiter]
    ENDLOOP                                                 ;^End Loop 3.

    if (_avcheck>0)                                         ;^Begin Condition 1: If current ATYPE in
Loop 2
                                                            ; has X>0 continue to report X. Else skip
ATYPE.
      _supertotal=0                                         ;^Initialize ATYPE total X.

      Print list= "Area Type ",_aat2(2.0), PRINTO=1         ;^Header
      Print list= "                                                    Number of Lanes per Direction
", PRINTO=1
      Print list= "FType          1           2           3           4           5           6
7           8           9       Totals", PRINTO=1
      Print list= "-----------------------------------------------------------------------------------
-----------------------------------------------", PRINTO=1

      LOOP _fiter=100,9900,100                              ;^Begin Loop 4: Cycles through FTYPE
                                                            ; by 1 in order to get two-digit FTYPE.
       _vcheck=0                                            ;^Initialize FTYPE X checking variable.

       LOOP _liter=1,9,1                                    ;^Begin Loop 5: Cycles through Lanes for
current
         _vcheck=_vcheck+_cntby[_aiter+_fiter+_liter]       ; FTYPE in Loop 4 and totals X checking
variable.
       ENDLOOP                                              ;^End Loop 5.

        if (_vcheck>0)                                      ;^Begin Condition 2: If current FTYPE in
Loop 4
```

```
           _fft2=int(_fiter/100)                          ; has X>0 continue to report X. Else skip
FTYPE.
           print list= _fft2(2.0),"    ", PRINTO=1
           _totvols=0                                      ;^Initialize FTYPE total X.

           LOOP _liter2=1,9,1                              ;^Begin Loop 6: Cycles through Lanes to
generate ATYPE by FTYPE by Lanes total X.
             print list="\\"," ",_cntby[_aiter+_fiter+_liter2](10.0C)," ", PRINTO=1
             _totvols=_totvols+_cntby[_aiter+_fiter+_liter2]
             _supertotal=_supertotal+_cntby[_aiter+_fiter+_liter2]
           ENDLOOP                                         ;^End Loop 6.

             print list="\\"," ",_totvols(10.0C), PRINTO=1
        endif                                              ;^End Condition 2.

      ENDLOOP                                              ;^End Loop 4.

      Print list= "-------------------------------------------------------------------------------
-------------------------------------------------", PRINTO=1
      print list="Totals", PRINTO=1

      LOOP _liter3=1,9,1                                  ;^Begin Loop 7: Cycles through Lanes for
                                                          ; current ATYPE in Loop 2.
        _lntotals=0                                       ;^Initialize Lane total X.

        LOOP _aiter2=_aiter,599999,100                    ;^Begin Loop 8: Cycles through FTYPE for
current ATYPE
          if (_aiter2>_aiter+9999) BREAK                  ; in Loop 2 to generate Lane total X.
          _lntotiter=_aiter2+_liter3
          _lntotals=_lntotals+_cntby[_lntotiter]
        ENDLOOP                                           ;^End Loop 8

        print list="\\"," ",_lntotals(10.0C)," ", PRINTO=1
      ENDLOOP                                             ;^End Loop 7

      print list="\\"," ",_supertotal(10.0C), PRINTO=1
      print list=" ", PRINTO=1
    endif                                                 ;^End Condition 1.

  ENDLOOP                                                 ;^End Loop 2.

  print list=" ", PRINTO=1
ENDLOOP                                                   ;^End Loop 1.

;-----------------2-DIGIT FACILITY TYPES BY TOTAL AREA TYPES--------------------

Print list= "Total Area Types ", PRINTO=1                 ;^Header
Print list= "                                                    Number of Lanes per Direction
", PRINTO=1
Print list= "FType         1           2           3           4           5           6           7
8           9       Totals", PRINTO=1
Print list= "-------------------------------------------------------------------------------------------
-----------------------------------------", PRINTO=1

LOOP _fiter2=100,9900,100                                 ;^Begin Loop 9: Cycles through FTYPES to
get
  _fft2=int(_fiter2/100)                                  ; two-digit FTYPE.

  _tafvcheck=0                                            ;^Initialize FTYPE X checking variable.

  LOOP _liter5=1,9,1                                        ;^Begin Loop 10: Cycles through Lanes
for current
                                                          ; FTYPE in Loop 9.
    LOOP _aiter4= 100000,599999,10000                       ;^Begin Loop 11: Cycles through ATYPE
for
      _tafvcheck=_tafvcheck+_cntby[_aiter4+_fiter2+_liter5] ; current Lanes and FTYPE in order to
total X checking variable.
    ENDLOOP                                                 ;^End Loop 11.

  ENDLOOP                                                 ;^End Loop 10.
```

```
  if (_tafvcheck>0)                                          ;^Begin Condition 3: If current FTYPE in
Loop 9
    print list= _fft2(2.0),"    ", PRINTO=1                 ; has X>0 continue to report X. Else skip
FTYPE.

    LOOP _liter4= 1,9,1                                        ;^Begin Loop 12: Cycles through Lanes
for current FTYPE
                                                             ; in Loop 9.
       _totftat=0                                           ;^Initialize FTYPE total X for all ATYPE.

      LOOP _aiter3= 100000,599999,10000                        ;^Begin Loop 13: Cycles through ATYPE
for current Lanes in Loop 12
       _totftat=_totftat+_cntby[_aiter3+_fiter2+_liter4] ; in order to generate total X for FTYPE
by Lane for all ATYPE.
      ENDLOOP                                                ;^End Loop 13.

      print list="\\"," ",_totftat(10.0C)," ", PRINTO=1
    ENDLOOP                                                  ;^End Loop 12.

    print list="\\"," ",_tafvcheck(10.0C), PRINTO=1
  endif                                                      ;^End Condition 3.

ENDLOOP                                                      ;^End Loop 9.

Print list= "-------------------------------------------------------------------------------------
-----------------------------------------", PRINTO=1
print list="Totals", PRINTO=1

_supertotal=0                                                ;^Initialize all ATYPE total X.

LOOP _liter6=1,9,1                                           ;^Begin Loop 14: Cycles through Lanes.

  _lntotals=0                                                ;^Initialize total X for Lanes.

  LOOP _aiter5=100000,599999,100                               ;^Begin Loop 15: Cycles through ATYPE
and
      _lntotiter=_aiter5+_liter6                            ; FTYPE in order to generate total X for
      _lntotals=_lntotals+_cntby[_lntotiter]                ; Lanes.
  ENDLOOP                                                    ;^End Loop 15.

  print list="\\"," ",_lntotals(10.0C)," ", PRINTO=1
  _supertotal=_supertotal+_lntotals                          ;^Generate total X for all ATYPE.

ENDLOOP                                                      ;^End Loop 14.
print list="\\"," ",_supertotal(10.0C), PRINTO=1
print list=" ","\n ", PRINTO=1

;---------------1-DIGIT FACILITY TYPES BY 1-DIGIT AREA TYPES SUMMARY----------------

Print list= "Total Summary Area Types by Facility Types ", PRINTO=1 ;^Header
Print list= "                                                    Single Digit Facility Types
", PRINTO=1
Print list= "AType         1x          2x          3x          4x          5x          6x          7x
8x          9x      Totals", PRINTO=1
Print list= "-------------------------------------------------------------------------------------
-----------------------------------------", PRINTO=1

LOOP _a1iter2=100000,599999,100000                           ;^Begin Loop 16: Cycles through ATYPE by
10 to
  _aat1=int(_a1iter2/100000)                                 ; get single digit ATYPE.
  print list= _aat1(1.0),"x","    ", PRINTO=1

  _fttotal=0                                                 ;^Initialize total X for all ATYPE

    LOOP _f1iter=1000,9900,1000                              ;^Begin Loop 17: Cycles through FTYPE by
10 to
                                                             ; get single digit FTYPE.
      _totftlns=0                                              ;^Initialize total X for all FTYPE by
all Lanes.
```

```
      LOOP _fiter3=_f1iter,9900,100                        ;^Begin Loop 18: Cycles through two-digit
FTYPE
        if (_fiter3>_f1iter+999) BREAK                      ; for current single digit FTYPE in Loop
17.

        LOOP _aiter6=_a1iter2,599999,10000                  ;^Begin Loop 19: Cycles through two-digit
ATYPE
          if (_aiter6>_a1iter2+99999) BREAK                 ; for current single digit ATYPE in Loop
16.

            LOOP _liter7=1,9,1                                          ;^Begin Loop 20: Cycles through
Lanes for current FTYPE and ATYPE
              _totftlns=_totftlns+_cntby[_aiter6+_fiter3+_liter7]   ; in order to generate total X
for FTYPE by ATYPE.
            ENDLOOP                                                      ;^End Loop 20.

        ENDLOOP                                             ;^End Loop 19.

      ENDLOOP                                               ;^End Loop 18.

      _fttotal=_fttotal+_totftlns                           ;^Generate total X for ATYPE.

      print list="\\"," ",_totftlns(10.0C)," ", PRINTO=1
    ENDLOOP                                                 ;^End Loop 17.

  print list="\\"," ",_fttotal(10.0c), PRINTO=1
ENDLOOP                                                     ;^End Loop 16.

Print list= "-------------------------------------------------------------------------------------------
-----------------------------------------", PRINTO=1
print list="Totals", PRINTO=1

_supertotal=0                                               ;^Initialize overall total X.

LOOP _f1iter2=1000,9900,1000                                ;^Begin Loop 21: Cycles through FTYPE by
10
                                                            ; to get single digit FTYPE.
  _ftotals=0                                                ;^Initialize total X by FTYPE

  LOOP _fiter4=_f1iter2,9900,100                            ;^Begin Loop 22: Cycles through FTYPE by
1 to
    if (_fiter4>_f1iter2+999) BREAK                           ; get all two-digit FTYPE for current
FTYPE in
                                                            ; Loop 21.
    LOOP _liter8=1,9,1                                      ;^Begin Loop 23: Cycles through Lanes.

      LOOP _aiter7=100000,599999,10000                      ;^Begin Loop 24: Cycles through ATYPE in
order
        _ftotiter=_aiter7+_fiter4+_liter8                     ; to generate total X by single digit
FTYPE.
        _ftotals=_ftotals+_cntby[_ftotiter]
      ENDLOOP                                               ;^End Loop 24.

    ENDLOOP                                                 ;^End Loop 23.

  ENDLOOP                                                   ;^End Loop 22.

  _supertotal=_supertotal+_ftotals                          ;^Generate overall total for all single
digit ATYPE
                                                            ; by all single digit FTYPE.
  print list="\\"," ",_ftotals(10.0C)," ", PRINTO=1
ENDLOOP                                                     ;^End Loop 21.

print list="\\"," ",_supertotal(10.0C), PRINTO=1
print list=" ", PRINTO=1
;*************************************************************************
; END COUNT REPORT
;*************************************************************************

;=========================================================================================
; BEGIN VOLUME/COUNT REPORT ------------ X = Volumes over Counts on Links w/ Counts
```

```
;=============================================================================================
Print list=" ", PRINTO=1
Print
list="*************************************************************************************************
***********************************", PRINTO=1
Print                                                                                          list="*
*", PRINTO=1
Print list="*                                          Volume over Count Ratios on Links with Counts
*", PRINTO=1
Print                                                                                          list="*
*", PRINTO=1
Print
list="*************************************************************************************************
***********************************", PRINTO=1
Print list=" ", PRINTO=1
;-----------------2-DIGIT FACILITY TYPES BY 2-DIGIT AREA TYPES--------------------

LOOP _a1iter=100000,599999,100000                          ;^Begin Loop 1: Cycles through Area Types
(ATYPE) by 10
  _aat1=int(_a1iter/100000)                                 ; in order to get single digit ATYPE.
  print list= "Area Type ",_aat1(1.0),"x Range:",
           "\n ", PRINTO=1

  LOOP _aiter=_a1iter,599999,10000                          ;^Begin Loop 2: Cycles through ATYPE by
1
    if (_aiter>_a1iter+99999) BREAK                         ; in order to get two-digit ATYPE.
    _aat2=int(_aiter/10000)

    _avcheck=0                                              ;^Initialize ATYPE X checking variable.

    LOOP _achkiter=_aiter,599999,1                          ;^Begin Loop 3: Cycles through Lanes and
Facility Types (FTYPE)
      if (_achkiter>_aiter+9999) BREAK                      ; for current ATYPE in Loop 2 and totals
X checking variable.
      _avcheck=_avcheck+_vcntby[_achkiter]
    ENDLOOP                                                 ;^End Loop 3.

    if (_avcheck>0)                                         ;^Begin Condition 1: If current ATYPE in
Loop 2
                                                            ; has X>0 continue to report X. Else skip
ATYPE.
      _supertotal=0                                         ;^Initialize ATYPE total X.
      _supercnts=0

      Print list= "Area Type ",_aat2(2.0), PRINTO=1         ;^Header
      Print list= "                                                    Number of Lanes per Direction
", PRINTO=1
      Print list= "FType          1           2           3           4           5           6
7           8           9       Totals", PRINTO=1
      Print list= "-------------------------------------------------------------------------------
-------------------------------------------------", PRINTO=1

      LOOP _fiter=100,9900,100                              ;^Begin Loop 4: Cycles through FTYPE
                                                            ; by 1 in order to get two-digit FTYPE.
       _vcheck=0                                            ;^Initialize FTYPE X checking variable.

       LOOP _liter=1,9,1                                    ;^Begin Loop 5: Cycles through Lanes for
current
         _vcheck=_vcheck+_vcntby[_aiter+_fiter+_liter]     ; FTYPE in Loop 4 and totals X checking
variable.
       ENDLOOP                                              ;^End Loop 5.

        if (_vcheck>0)                                      ;^Begin Condition 2: If current FTYPE in
Loop 4
          _fft2=int(_fiter/100)                            ; has X>0 continue to report X. Else skip
FTYPE.
          print list= _fft2(2.0),"    ", PRINTO=1
          _totvols=0                                        ;^Initialize FTYPE total X.
          _totcnts=0
```

```
            LOOP _liter2=1,9,1                              ;^Begin Loop 6: Cycles through Lanes to
generate ATYPE by FTYPE by Lanes total X.
             if (_cntby[_aiter+_fiter+_liter2]>0)
               _links=_volby[_aiter+_fiter+_liter2]/_cntby[_aiter+_fiter+_liter2]
             else
               _links=0
             endif
             print list="\\"," ",_links(10.2C)," ", PRINTO=1
             _totvols=_totvols+_volby[_aiter+_fiter+_liter2]
             _totcnts=_totcnts+_cntby[_aiter+_fiter+_liter2]
             _supertotal=_supertotal+_volby[_aiter+_fiter+_liter2]
             _supercnts=_supercnts+_cntby[_aiter+_fiter+_liter2]
           ENDLOOP                                          ;^End Loop 6.

             if (_totcnts>0)
               _totvc=_totvols/_totcnts
             else
               _totvc=0
             endif
           print list="\\"," ",_totvc(10.2C), PRINTO=1
        endif                                               ;^End Condition 2.

      ENDLOOP                                               ;^End Loop 4.

      Print list= "-------------------------------------------------------------------------------
-------------------------------------------------", PRINTO=1
      print list="Totals", PRINTO=1

      LOOP _liter3=1,9,1                                   ;^Begin Loop 7: Cycles through Lanes for
                                                           ; current ATYPE in Loop 2.
        _lntotals=0                                        ;^Initialize Lane total X.
        _lncnts=0

        LOOP _aiter2=_aiter,599999,100                     ;^Begin Loop 8: Cycles through FTYPE for
current ATYPE
          if (_aiter2>_aiter+9999) BREAK                   ; in Loop 2 to generate Lane total X.
          _lntotiter=_aiter2+_liter3
          _lntotals=_lntotals+_volby[_lntotiter]
          _lncnts=_lncnts+_cntby[_lntotiter]
        ENDLOOP                                            ;^End Loop 8

        if (_lncnts>0)
          _lnvc=_lntotals/_lncnts
        else
          _lnvc=0
        endif
        print list="\\"," ",_lnvc(10.2C)," ", PRINTO=1
      ENDLOOP                                              ;^End Loop 7

      if (_supercnts>0)
        _supervc=_supertotal/_supercnts
      else
        _supervc=0
      endif
      print list="\\"," ",_supervc(10.2C), PRINTO=1
      print list=" ", PRINTO=1
    endif                                                  ;^End Condition 1.

  ENDLOOP                                                  ;^End Loop 2.

  print list=" ", PRINTO=1
ENDLOOP                                                    ;^End Loop 1.

;-----------------2-DIGIT FACILITY TYPES BY TOTAL AREA TYPES--------------------

Print list= "Total Area Types ", PRINTO=1                  ;^Header
Print list= "                                                        Number of Lanes per Direction
", PRINTO=1
Print list= "FType         1          2          3          4          5          6          7
8          9       Totals", PRINTO=1
```

```
Print list= "-------------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1

LOOP _fiter2=100,9900,100                                    ;^Begin Loop 9: Cycles through FTYPES to
get
  _fft2=int(_fiter2/100)                                     ; two-digit FTYPE.

  _tafvcheck=0                                               ;^Initialize FTYPE X checking variable.
  _tafcnts=0

  LOOP _liter5=1,9,1                                           ;^Begin Loop 10: Cycles through Lanes
for current
                                                             ; FTYPE in Loop 9.
    LOOP _aiter4= 100000,599999,10000                          ;^Begin Loop 11: Cycles through ATYPE
for
      _tafvcheck=_tafvcheck+_volby[_aiter4+_fiter2+_liter5] ; current Lanes and FTYPE in order to
total X checking variable.
      _tafcnts=_tafcnts+_cntby[_aiter4+_fiter2+_liter5]
    ENDLOOP                                                    ;^End Loop 11.

  ENDLOOP                                                    ;^End Loop 10.

  if (_tafvcheck>0)                                          ;^Begin Condition 3: If current FTYPE in
Loop 9
    print list= _fft2(2.0),"    ", PRINTO=1                 ; has X>0 continue to report X. Else skip
FTYPE.

    LOOP _liter4= 1,9,1                                        ;^Begin Loop 12: Cycles through Lanes
for current FTYPE
                                                             ; in Loop 9.
        _totftat=0                                           ;^Initialize FTYPE total X for all ATYPE.
        _totcnts=0

      LOOP _aiter3= 100000,599999,10000                        ;^Begin Loop 13: Cycles through ATYPE
for current Lanes in Loop 12
        _totftat=_totftat+_volby[_aiter3+_fiter2+_liter4] ; in order to generate total X for FTYPE
by Lane for all ATYPE.
        _totcnts=_totcnts+_cntby[_aiter3+_fiter2+_liter4]
      ENDLOOP                                                ;^End Loop 13.
      if (_totcnts>0)
        _totvc=_totftat/_totcnts
      else
        _totvc=0
      endif
      print list="\\"," ",_totvc(10.2C)," ", PRINTO=1
    ENDLOOP                                                  ;^End Loop 12.
    if (_tafcnts>0)
      _tafvc=_tafvcheck/_tafcnts
    else
      _tafvc=0
    endif
    print list="\\"," ",_tafvc(10.2C), PRINTO=1
  endif                                                      ;^End Condition 3.

ENDLOOP                                                      ;^End Loop 9.

Print list= "-------------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1
print list="Totals", PRINTO=1

_supertotal=0                                                ;^Initialize all ATYPE total X.
_supercnts=0

LOOP _liter6=1,9,1                                           ;^Begin Loop 14: Cycles through Lanes.

  _lntotals=0                                                ;^Initialize total X for Lanes.
  _lncnts=0

  LOOP _aiter5=100000,599999,100                               ;^Begin Loop 15: Cycles through ATYPE
and
      _lntotiter=_aiter5+_liter6                             ; FTYPE in order to generate total X for
```

```
      _lntotals=_lntotals+_volby[_lntotiter]              ; Lanes.
      _lncnts=_lncnts+_cntby[_lntotiter]
  ENDLOOP                                                 ;^End Loop 15.
  if (_lncnts>0)
    _lnvc=_lntotals/_lncnts
  else
    _lnvc=0
  endif
  print list="\\"," ",_lnvc(10.2C)," ", PRINTO=1
  _supertotal=_supertotal+_lntotals                       ;^Generate total X for all ATYPE.
  _supercnts=_supercnts+_lncnts
ENDLOOP                                                   ;^End Loop 14.
if (_supercnts>0)
  _supervc=_supertotal/_supercnts
else
  _supervc=0
endif
print list="\\"," ",_supervc(10.2C), PRINTO=1
print list=" ","\n ", PRINTO=1

;---------------1-DIGIT FACILITY TYPES BY 1-DIGIT AREA TYPES SUMMARY----------------

Print list= "Total Summary Area Types by Facility Types ", PRINTO=1 ;^Header
Print list= "                                                        Single Digit Facility Types
", PRINTO=1
Print list= "AType         1x          2x          3x          4x          5x          6x          7x
8x          9x      Totals", PRINTO=1
Print list= "-------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1

LOOP _a1iter2=100000,599999,100000                        ;^Begin Loop 16: Cycles through ATYPE by
10 to
  _aat1=int(_a1iter2/100000)                              ; get single digit ATYPE.
  print list= _aat1(1.0),"x","    ", PRINTO=1

  _fttotal=0                                              ;^Initialize total X for all ATYPE
  _ftcnts=0

    LOOP _f1iter=1000,9900,1000                           ;^Begin Loop 17: Cycles through FTYPE by
10 to
                                                          ; get single digit FTYPE.
      _totftlns=0                                           ;^Initialize total X for all FTYPE by
all Lanes.
      _totftcnts=0

      LOOP _fiter3=_f1iter,9900,100                       ;^Begin Loop 18: Cycles through two-digit
FTYPE
        if (_fiter3>_f1iter+999) BREAK                     ; for current single digit FTYPE in Loop
17.

        LOOP _aiter6=_a1iter2,599999,10000                ;^Begin Loop 19: Cycles through two-digit
ATYPE
          if (_aiter6>_a1iter2+99999) BREAK                ; for current single digit ATYPE in Loop
16.

            LOOP _liter7=1,9,1                                        ;^Begin Loop 20: Cycles through
Lanes for current FTYPE and ATYPE
              _totftlns=_totftlns+_volby[_aiter6+_fiter3+_liter7]   ; in order to generate total X
for FTYPE by ATYPE.
              _totftcnts=_totftcnts+_cntby[_aiter6+_fiter3+_liter7]
            ENDLOOP                                                    ;^End Loop 20.

        ENDLOOP                                           ;^End Loop 19.

      ENDLOOP                                             ;^End Loop 18.

      _fttotal=_fttotal+_totftlns                         ;^Generate total X for ATYPE.
      _ftcnts=_ftcnts+_totftcnts

      if (_totftcnts>0)
        _totftvc=_totftlns/_totftcnts
```

```
      else
        _totftvc=0
      endif
      print list="\\"," ",_totftvc(10.2C)," ", PRINTO=1
    ENDLOOP                                                ;^End Loop 17.
    if (_ftcnts>0)
      _ftvc=_fttotal/_ftcnts
    else
      _ftvc=0
    endif
  print list="\\"," ",_ftvc(10.2c), PRINTO=1
ENDLOOP                                                    ;^End Loop 16.

Print list= "-------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1
print list="Totals", PRINTO=1

_supertotal=0                                              ;^Initialize overall total X.
_supercnts=0

LOOP _f1iter2=1000,9900,1000                               ;^Begin Loop 21: Cycles through FTYPE by
10
                                                           ; to get single digit FTYPE.
  _ftotals=0                                               ;^Initialize total X by FTYPE
  _ftcnts=0

  LOOP _fiter4=_f1iter2,9900,100                           ;^Begin Loop 22: Cycles through FTYPE by
1 to
    if (_fiter4>_f1iter2+999) BREAK                          ; get all two-digit FTYPE for current
FTYPE in
                                                           ; Loop 21.
    LOOP _liter8=1,9,1                                     ;^Begin Loop 23: Cycles through Lanes.

      LOOP _aiter7=100000,599999,10000                     ;^Begin Loop 24: Cycles through ATYPE in
order
        _ftotiter=_aiter7+_fiter4+_liter8                    ; to generate total X by single digit
FTYPE.
        _ftotals=_ftotals+_volby[_ftotiter]
        _ftcnts=_ftcnts+_cntby[_ftotiter]
      ENDLOOP                                              ;^End Loop 24.

    ENDLOOP                                                ;^End Loop 23.

  ENDLOOP                                                  ;^End Loop 22.

  _supertotal=_supertotal+_ftotals                          ;^Generate overall total for all single
digit ATYPE
  _supercnts=_supercnts+_ftcnts                           ; by all single digit FTYPE.

  if (_ftcnts>0)
    _ftvc=_ftotals/_ftcnts
  else
    _ftvc=0
  endif

  print list="\\"," ",_ftvc(10.2C)," ", PRINTO=1
ENDLOOP                                                    ;^End Loop 21.
if (_supercnts>0)
  _supervc=_supertotal/_supercnts
else
  _supervc=0
endif
print list="\\"," ",_supervc(10.2C), PRINTO=1
print list=" ", PRINTO=1
;*************************************************************************
; END VOLUME OVER COUNT REPORT
;*************************************************************************

;=========================================================================================
; BEGIN VOLUME ON ALL LINKS ------------ X = VOLUME
;=========================================================================================
```

```
Print list=" ", PRINTO=1
Print
list="*************************************************************************************************
***********************************", PRINTO=1
Print                                                                                              list="*
*", PRINTO=1
Print list="*                                        Total Volume on All Links (Centroid Connectors
Excluded)                                        *", PRINTO=1
Print                                                                                              list="*
*", PRINTO=1
Print
list="*************************************************************************************************
***********************************", PRINTO=1
Print list=" ", PRINTO=1
;-----------------2-DIGIT FACILITY TYPES BY 2-DIGIT AREA TYPES--------------------

LOOP _a1iter=100000,599999,100000                          ;^Begin Loop 1: Cycles through Area Types
(ATYPE) by 10
  _aat1=int(_a1iter/100000)                                 ; in order to get single digit ATYPE.
  print list= "Area Type ",_aat1(1.0),"x Range:",
           "\n ", PRINTO=1

  LOOP _aiter=_a1iter,599999,10000                          ;^Begin Loop 2: Cycles through ATYPE by
1
    if (_aiter>_a1iter+99999) BREAK                         ; in order to get two-digit ATYPE.
    _aat2=int(_aiter/10000)

    _avcheck=0                                              ;^Initialize ATYPE X checking variable.

    LOOP _achkiter=_aiter,599999,1                          ;^Begin Loop 3: Cycles through Lanes and
Facility Types (FTYPE)
      if (_achkiter>_aiter+9999) BREAK                      ; for current ATYPE in Loop 2 and totals
X checking variable.
      _avcheck=_avcheck+_volall[_achkiter]
    ENDLOOP                                                 ;^End Loop 3.

    if (_avcheck>0)                                         ;^Begin Condition 1: If current ATYPE in
Loop 2
                                                            ; has X>0 continue to report X. Else skip
ATYPE.
      _supertotal=0                                         ;^Initialize ATYPE total X.

      Print list= "Area Type ",_aat2(2.0), PRINTO=1         ;^Header
      Print list= "                                                    Number of Lanes per Direction
", PRINTO=1
      Print list= "FType          1           2           3           4           5           6
7           8           9       Totals", PRINTO=1
      Print list= "-----------------------------------------------------------------------------------
-------------------------------------------------", PRINTO=1

      LOOP _fiter=100,9900,100                              ;^Begin Loop 4: Cycles through FTYPE
                                                            ; by 1 in order to get two-digit FTYPE.
       _vcheck=0                                            ;^Initialize FTYPE X checking variable.

       LOOP _liter=1,9,1                                    ;^Begin Loop 5: Cycles through Lanes for
current
         _vcheck=_vcheck+_volall[_aiter+_fiter+_liter]       ; FTYPE in Loop 4 and totals X checking
variable.
       ENDLOOP                                              ;^End Loop 5.

        if (_vcheck>0 & (_fiter<5000 | _fiter>5999))        ;^Begin Condition 2: If current FTYPE in
Loop 4
           _fft2=int(_fiter/100)                           ; has X>0 continue to report X. Else skip
FTYPE.
           print list= _fft2(2.0),"    ", PRINTO=1
           _totvols=0                                       ;^Initialize FTYPE total X.

           LOOP _liter2=1,9,1                               ;^Begin Loop 6: Cycles through Lanes to
generate ATYPE by FTYPE by Lanes total X.
             print list="\\"," ",_volall[_aiter+_fiter+_liter2](10.0C)," ", PRINTO=1
             _totvols=_totvols+_volall[_aiter+_fiter+_liter2]
```

```
            _supertotal=_supertotal+_volall[_aiter+_fiter+_liter2]
          ENDLOOP                                          ;^End Loop 6.

            print list="\\"," ",_totvols(10.0C), PRINTO=1
        endif                                              ;^End Condition 2.

      ENDLOOP                                              ;^End Loop 4.

      Print list= "-------------------------------------------------------------------------------
-----------------------------------------------", PRINTO=1
      print list="Totals", PRINTO=1

      LOOP _liter3=1,9,1                                   ;^Begin Loop 7: Cycles through Lanes for
                                                           ; current ATYPE in Loop 2.
        _lntotals=0                                        ;^Initialize Lane total X.

        LOOP _aiter2=_aiter,599999,100                     ;^Begin Loop 8: Cycles through FTYPE for
current ATYPE
          if (_aiter2>_aiter+9999) BREAK                   ; in Loop 2 to generate Lane total X.
          if (_aiter2<_aiter+5000 | _aiter2>_aiter+5999)
            _lntotiter=_aiter2+_liter3
            _lntotals=_lntotals+_volall[_lntotiter]
          endif
        ENDLOOP                                            ;^End Loop 8

        print list="\\"," ",_lntotals(10.0C)," ", PRINTO=1
      ENDLOOP                                              ;^End Loop 7

      print list="\\"," ",_supertotal(10.0C), PRINTO=1
      print list=" ", PRINTO=1
    endif                                                  ;^End Condition 1.

  ENDLOOP                                                  ;^End Loop 2.

  print list=" ", PRINTO=1
ENDLOOP                                                    ;^End Loop 1.

;-----------------2-DIGIT FACILITY TYPES BY TOTAL AREA TYPES--------------------

Print list= "Total Area Types ", PRINTO=1                  ;^Header
Print list= "                                                        Number of Lanes per Direction
", PRINTO=1
Print list= "FType         1           2           3           4           5           6           7
8           9       Totals", PRINTO=1
Print list= "-------------------------------------------------------------------------------------
-----------------------------------------", PRINTO=1

LOOP _fiter2=100,9900,100                                  ;^Begin Loop 9: Cycles through FTYPES to
get
  _fft2=int(_fiter2/100)                                   ; two-digit FTYPE.

  _tafvcheck=0                                             ;^Initialize FTYPE X checking variable.
 if (_fft2<50 | _fft2>59)
  LOOP _liter5=1,9,1                                         ;^Begin Loop 10: Cycles through Lanes
for current
                                                           ; FTYPE in Loop 9.
    LOOP _aiter4= 100000,599999,10000                        ;^Begin Loop 11: Cycles through ATYPE
for
      _tafvcheck=_tafvcheck+_volall[_aiter4+_fiter2+_liter5] ; current Lanes and FTYPE in order to
total X checking variable.
    ENDLOOP                                                  ;^End Loop 11.

  ENDLOOP                                                  ;^End Loop 10.

  if (_tafvcheck>0)                                        ;^Begin Condition 3: If current FTYPE in
Loop 9
    print list= _fft2(2.0),"    ", PRINTO=1              ; has X>0 continue to report X. Else skip
FTYPE.

    LOOP _liter4= 1,9,1                                      ;^Begin Loop 12: Cycles through Lanes
for current FTYPE
```

```
                                                          ; in Loop 9.
        _totftat=0                                       ;^Initialize FTYPE total X for all ATYPE.

      LOOP _aiter3= 100000,599999,10000                    ;^Begin Loop 13: Cycles through ATYPE
for current Lanes in Loop 12
        _totftat=_totftat+_volall[_aiter3+_fiter2+_liter4] ; in order to generate total X for FTYPE
by Lane for all ATYPE.
      ENDLOOP                                            ;^End Loop 13.

      print list="\\"," ",_totftat(10.0C)," ", PRINTO=1
    ENDLOOP                                              ;^End Loop 12.

    print list="\\"," ",_tafvcheck(10.0C), PRINTO=1
  endif                                                  ;^End Condition 3.
 endif
ENDLOOP                                                  ;^End Loop 9.

Print list= "-----------------------------------------------------------------------------------
-----------------------------------------", PRINTO=1
print list="Totals", PRINTO=1

_supertotal=0                                            ;^Initialize all ATYPE total X.

LOOP _liter6=1,9,1                                       ;^Begin Loop 14: Cycles through Lanes.

  _lntotals=0                                            ;^Initialize total X for Lanes.

  LOOP _aiter5=100000,599999,100                           ;^Begin Loop 15: Cycles through ATYPE
and
    if ((_aiter5<105000 | _aiter5>105999) &
        (_aiter5<115000 | _aiter5>115999) &
        (_aiter5<125000 | _aiter5>125999) &
        (_aiter5<135000 | _aiter5>135999) &
        (_aiter5<145000 | _aiter5>145999) &
        (_aiter5<155000 | _aiter5>155999) &
        (_aiter5<165000 | _aiter5>165999) &
        (_aiter5<175000 | _aiter5>175999) &
        (_aiter5<185000 | _aiter5>185999) &
        (_aiter5<195000 | _aiter5>195999) &
        (_aiter5<205000 | _aiter5>205999) &
        (_aiter5<215000 | _aiter5>215999) &
        (_aiter5<225000 | _aiter5>225999) &
        (_aiter5<235000 | _aiter5>235999) &
        (_aiter5<245000 | _aiter5>245999) &
        (_aiter5<255000 | _aiter5>255999) &
        (_aiter5<265000 | _aiter5>265999) &
        (_aiter5<275000 | _aiter5>275999) &
        (_aiter5<285000 | _aiter5>285999) &
        (_aiter5<295000 | _aiter5>295999) &
        (_aiter5<305000 | _aiter5>305999) &
        (_aiter5<315000 | _aiter5>315999) &
        (_aiter5<325000 | _aiter5>325999) &
        (_aiter5<335000 | _aiter5>335999) &
        (_aiter5<345000 | _aiter5>345999) &
        (_aiter5<355000 | _aiter5>355999) &
        (_aiter5<365000 | _aiter5>365999) &
        (_aiter5<375000 | _aiter5>375999) &
        (_aiter5<385000 | _aiter5>385999) &
        (_aiter5<395000 | _aiter5>395999) &
        (_aiter5<405000 | _aiter5>405999) &
        (_aiter5<415000 | _aiter5>415999) &
        (_aiter5<425000 | _aiter5>425999) &
        (_aiter5<435000 | _aiter5>435999) &
        (_aiter5<445000 | _aiter5>445999) &
        (_aiter5<455000 | _aiter5>455999) &
        (_aiter5<465000 | _aiter5>465999) &
        (_aiter5<475000 | _aiter5>475999) &
        (_aiter5<485000 | _aiter5>485999) &
        (_aiter5<495000 | _aiter5>495999) &
        (_aiter5<505000 | _aiter5>505999) &
        (_aiter5<515000 | _aiter5>515999) &
```

```
        (_aiter5<525000 | _aiter5>525999) &
        (_aiter5<535000 | _aiter5>535999) &
        (_aiter5<545000 | _aiter5>545999) &
        (_aiter5<555000 | _aiter5>555999) &
        (_aiter5<565000 | _aiter5>565999) &
        (_aiter5<575000 | _aiter5>575999) &
        (_aiter5<585000 | _aiter5>585999) &
        (_aiter5<595000 | _aiter5>595999))

      _lntotiter=_aiter5+_liter6                          ; FTYPE in order to generate total X for
      _lntotals=_lntotals+_volall[_lntotiter]              ; Lanes.

    endif
  ENDLOOP                                                 ;^End Loop 15.

  print list="\\"," ",_lntotals(10.0C)," ", PRINTO=1
  _supertotal=_supertotal+_lntotals                       ;^Generate total X for all ATYPE.

ENDLOOP                                                   ;^End Loop 14.
print list="\\"," ",_supertotal(10.0C), PRINTO=1
print list=" ","\n ", PRINTO=1

;---------------1-DIGIT FACILITY TYPES BY 1-DIGIT AREA TYPES SUMMARY----------------

Print list= "Total Summary Area Types by Facility Types ", PRINTO=1 ;^Header
Print list= "                                                    Single Digit Facility Types
", PRINTO=1
Print list= "AType         1x          2x          3x          4x          5x          6x          7x
8x          9x      Totals", PRINTO=1
Print list= "-------------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1

LOOP _a1iter2=100000,599999,100000                        ;^Begin Loop 16: Cycles through ATYPE by
10 to
  _aat1=int(_a1iter2/100000)                              ; get single digit ATYPE.
  print list= _aat1(1.0),"x","    ", PRINTO=1

  _fttotal=0                                              ;^Initialize total X for all ATYPE

    LOOP _f1iter=1000,9900,1000                           ;^Begin Loop 17: Cycles through FTYPE by
10 to
                                                          ; get single digit FTYPE.
      _totftlns=0                                           ;^Initialize total X for all FTYPE by
all Lanes.
      if (_f1iter<5000 | _f1iter>5999)
      LOOP _fiter3=_f1iter,9900,100                       ;^Begin Loop 18: Cycles through two-digit
FTYPE
        if (_fiter3>_f1iter+999) BREAK                    ; for current single digit FTYPE in Loop
17.

        LOOP _aiter6=_a1iter2,599999,10000                ;^Begin Loop 19: Cycles through two-digit
ATYPE
          if (_aiter6>_a1iter2+99999) BREAK               ; for current single digit ATYPE in Loop
16.

            LOOP _liter7=1,9,1                                ;^Begin Loop 20: Cycles through
Lanes for current FTYPE and ATYPE
              _totftlns=_totftlns+_volall[_aiter6+_fiter3+_liter7]   ; in order to generate total
X for FTYPE by ATYPE.
            ENDLOOP                                             ;^End Loop 20.

        ENDLOOP                                           ;^End Loop 19.

      ENDLOOP                                             ;^End Loop 18.
      endif
      _fttotal=_fttotal+_totftlns                         ;^Generate total X for ATYPE.

      print list="\\"," ",_totftlns(10.0C)," ", PRINTO=1
    ENDLOOP                                               ;^End Loop 17.

  print list="\\"," ",_fttotal(10.0c), PRINTO=1
```

```
ENDLOOP                                                 ;^End Loop 16.

Print list= "-----------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1
print list="Totals", PRINTO=1

_supertotal=0                                           ;^Initialize overall total X.

LOOP _f1iter2=1000,9900,1000                            ;^Begin Loop 21: Cycles through FTYPE by
10
                                                        ; to get single digit FTYPE.
  _ftotals=0                                            ;^Initialize total X by FTYPE

  LOOP _fiter4=_f1iter2,9900,100                        ;^Begin Loop 22: Cycles through FTYPE by
1 to
    if (_fiter4>_f1iter2+999) BREAK                       ; get all two-digit FTYPE for current
FTYPE in
    if (_f1iter2<5000 | _f1iter2>5999)                                                  ; Loop
21.
    LOOP _liter8=1,9,1                                  ;^Begin Loop 23: Cycles through Lanes.

      LOOP _aiter7=100000,599999,10000                  ;^Begin Loop 24: Cycles through ATYPE in
order
        _ftotiter=_aiter7+_fiter4+_liter8                 ; to generate total X by single digit
FTYPE.
        _ftotals=_ftotals+_volall[_ftotiter]
      ENDLOOP                                           ;^End Loop 24.

    ENDLOOP                                             ;^End Loop 23.
    endif
  ENDLOOP                                               ;^End Loop 22.
  _supertotal=_supertotal+_ftotals                      ;^Generate overall total for all single
digit ATYPE
                                                        ; by all single digit FTYPE.
  print list="\\"," ",_ftotals(10.0C)," ", PRINTO=1
ENDLOOP                                                 ;^End Loop 21.
_totalvolumes=_supertotal
print list="\\"," ",_supertotal(10.0C), PRINTO=1
print list=" ", PRINTO=1
;*********************************************************************
; END VOLUME ON ALL LINKS REPORT
;*********************************************************************

;=============================================================================================
; BEGIN VOLUME PERCENTAGES ON ALL LINKS ------------- X = VOLUME
;=============================================================================================
Print list=" ", PRINTO=1
Print
list="*************************************************************************************
***********************************", PRINTO=1
Print                                                                                 list="*
*", PRINTO=1
Print list="*                                 Volume Percentages on All Links (Centroid Connectors
Excluded)                                    *", PRINTO=1
Print                                                                                 list="*
*", PRINTO=1
Print
list="*************************************************************************************
***********************************", PRINTO=1
Print list=" ", PRINTO=1
;-----------------2-DIGIT FACILITY TYPES BY 2-DIGIT AREA TYPES--------------------
_supersuper=0
LOOP _supera= 100000,599999,10000                       ;^Begin Loops Pre-1 through Pre-3: Cycles
through all non-centroid
  LOOP _superf= 1000,9999,100                             ; connector links to generate overall
total X.
   if (_superf<5000 | _superf>5999)
     LOOP _superl=1,9,1
       _supersuper=_supersuper+_volall[_supera+_superf+_superl]/100   ;^Divide   by   100   to   get
pecentages and not ratios in later computations.
     ENDLOOP
```

```
   endif
  ENDLOOP                                                  ;End Loops Pre-3 through Pre-1.
ENDLOOP

LOOP _a1iter=100000,599999,100000                         ;^Begin Loop 1: Cycles through Area Types
(ATYPE) by 10
  _aat1=int(_a1iter/100000)                                ; in order to get single digit ATYPE.
  print list= "Area Type ",_aat1(1.0),"x Range:",
           "\n ", PRINTO=1

  LOOP _aiter=_a1iter,599999,10000                         ;^Begin Loop 2: Cycles through ATYPE by
1
    if (_aiter>_a1iter+99999) BREAK                        ; in order to get two-digit ATYPE.
    _aat2=int(_aiter/10000)

    _avcheck=0                                             ;^Initialize ATYPE X checking variable.



    LOOP _achkiter=_aiter,599999,1                         ;^Begin Loop 3: Cycles through Lanes and
Facility Types (FTYPE)
      if (_achkiter>_aiter+9999) BREAK                     ; for current ATYPE in Loop 2 and totals
X checking variable.
      _avcheck=_avcheck+_volall[_achkiter]
    ENDLOOP                                                ;^End Loop 3.

    if (_avcheck>0)                                        ;^Begin Condition 1: If current ATYPE in
Loop 2
                                                           ; has X>0 continue to report X. Else skip
ATYPE.
      _supertotal=0                                        ;^Initialize ATYPE total X.

      Print list= "Area Type ",_aat2(2.0), PRINTO=1        ;^Header
      Print list= "                                                  Number of Lanes per Direction
", PRINTO=1
      Print list= "FType          1           2           3           4           5           6
7           8           9       Totals", PRINTO=1
      Print list= "-----------------------------------------------------------------------------------
-----------------------------------------", PRINTO=1

      LOOP _fiter=100,9900,100                             ;^Begin Loop 4: Cycles through FTYPE
                                                           ; by 1 in order to get two-digit FTYPE.
       _vcheck=0                                           ;^Initialize FTYPE X checking variable.

       LOOP _liter=1,9,1                                   ;^Begin Loop 5: Cycles through Lanes for
current
         _vcheck=_vcheck+_volall[_aiter+_fiter+_liter]     ; FTYPE in Loop 4 and totals X checking
variable.
       ENDLOOP                                             ;^End Loop 5.

        if (_vcheck>0 & (_fiter<5000 | _fiter>5999))       ;^Begin Condition 2: If current FTYPE in
Loop 4
          _fft2=int(_fiter/100)                            ; has X>0 continue to report X. Else skip
FTYPE.
          print list= _fft2(2.0),"    ", PRINTO=1
          _totvols=0                                       ;^Initialize FTYPE total X.

          LOOP _liter2=1,9,1                               ;^Begin Loop 6: Cycles through Lanes to
generate ATYPE by FTYPE by Lanes total X.
            print list="\\"," ",(_volall[_aiter+_fiter+_liter2]/_supersuper)(10.2C)," ", PRINTO=1
            _totvols=_totvols+_volall[_aiter+_fiter+_liter2]
            _supertotal=_supertotal+_volall[_aiter+_fiter+_liter2]
          ENDLOOP                                          ;^End Loop 6.

            print list="\\"," ",(_totvols/_supersuper)(10.2C), PRINTO=1
        endif                                              ;^End Condition 2.

      ENDLOOP                                              ;^End Loop 4.

      Print list= "-----------------------------------------------------------------------------------
-----------------------------------------", PRINTO=1
```

```
      print list="Totals", PRINTO=1

      LOOP _liter3=1,9,1                                    ;^Begin Loop 7: Cycles through Lanes for
                                                            ; current ATYPE in Loop 2.
        _lntotals=0                                         ;^Initialize Lane total X.

        LOOP _aiter2=_aiter,599999,100                      ;^Begin Loop 8: Cycles through FTYPE for
current ATYPE
          if (_aiter2>_aiter+9999) BREAK                    ; in Loop 2 to generate Lane total X.
          if (_aiter2<_aiter+5000 | _aiter2>_aiter+5999)
            _lntotiter=_aiter2+_liter3
            _lntotals=_lntotals+_volall[_lntotiter]
          endif
        ENDLOOP                                             ;^End Loop 8

        print list="\\"," ",(_lntotals/_supersuper)(10.2C)," ", PRINTO=1
      ENDLOOP                                               ;^End Loop 7

      print list="\\"," ",(_supertotal/_supersuper)(10.2C), PRINTO=1
      print list=" ", PRINTO=1
    endif                                                   ;^End Condition 1.

  ENDLOOP                                                   ;^End Loop 2.

  print list=" ", PRINTO=1
ENDLOOP                                                     ;^End Loop 1.

;-----------------2-DIGIT FACILITY TYPES BY TOTAL AREA TYPES--------------------

Print list= "Total Area Types ", PRINTO=1                   ;^Header
Print list= "                                                        Number of Lanes per Direction
", PRINTO=1
Print list= "FType         1           2           3           4           5           6           7
8           9       Totals", PRINTO=1
Print list= "-------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1

LOOP _fiter2=100,9900,100                                   ;^Begin Loop 9: Cycles through FTYPES to
get
  _fft2=int(_fiter2/100)                                    ; two-digit FTYPE.

  _tafvcheck=0                                              ;^Initialize FTYPE X checking variable.
 if (_fft2<50 | _fft2>59)
  LOOP _liter5=1,9,1                                          ;^Begin Loop 10: Cycles through Lanes
for current
                                                            ; FTYPE in Loop 9.
    LOOP _aiter4= 100000,599999,10000                         ;^Begin Loop 11: Cycles through ATYPE
for
      _tafvcheck=_tafvcheck+_volall[_aiter4+_fiter2+_liter5] ; current Lanes and FTYPE in order to
total X checking variable.
    ENDLOOP                                                   ;^End Loop 11.

  ENDLOOP                                                   ;^End Loop 10.

  if (_tafvcheck>0)                                         ;^Begin Condition 3: If current FTYPE in
Loop 9
    print list= _fft2(2.0),"    ", PRINTO=1                 ; has X>0 continue to report X. Else skip
FTYPE.

    LOOP _liter4= 1,9,1                                       ;^Begin Loop 12: Cycles through Lanes
for current FTYPE
                                                            ; in Loop 9.
        _totftat=0                                          ;^Initialize FTYPE total X for all ATYPE.

      LOOP _aiter3= 100000,599999,10000                       ;^Begin Loop 13: Cycles through ATYPE
for current Lanes in Loop 12
        _totftat=_totftat+_volall[_aiter3+_fiter2+_liter4] ; in order to generate total X for FTYPE
by Lane for all ATYPE.
      ENDLOOP                                               ;^End Loop 13.

      print list="\\"," ",(_totftat/_supersuper)(10.2C)," ", PRINTO=1
```

```
    ENDLOOP                                                 ;^End Loop 12.

    print list="\\"," ",(_tafvcheck/_supersuper)(10.2C), PRINTO=1
  endif                                                     ;^End Condition 3.
 endif
ENDLOOP                                                     ;^End Loop 9.

Print list= "-------------------------------------------------------------------------------------------
-----------------------------------------", PRINTO=1
print list="Totals", PRINTO=1

_supertotal=0                                               ;^Initialize all ATYPE total X.

LOOP _liter6=1,9,1                                          ;^Begin Loop 14: Cycles through Lanes.


  _lntotals=0                                               ;^Initialize total X for Lanes.

  LOOP _aiter5=100000,599999,100                              ;^Begin Loop 15: Cycles through ATYPE
and
    if ((_aiter5<105000 | _aiter5>105999) &
        (_aiter5<115000 | _aiter5>115999) &
        (_aiter5<125000 | _aiter5>125999) &
        (_aiter5<135000 | _aiter5>135999) &
        (_aiter5<145000 | _aiter5>145999) &
        (_aiter5<155000 | _aiter5>155999) &
        (_aiter5<165000 | _aiter5>165999) &
        (_aiter5<175000 | _aiter5>175999) &
        (_aiter5<185000 | _aiter5>185999) &
        (_aiter5<195000 | _aiter5>195999) &
        (_aiter5<205000 | _aiter5>205999) &
        (_aiter5<215000 | _aiter5>215999) &
        (_aiter5<225000 | _aiter5>225999) &
        (_aiter5<235000 | _aiter5>235999) &
        (_aiter5<245000 | _aiter5>245999) &
        (_aiter5<255000 | _aiter5>255999) &
        (_aiter5<265000 | _aiter5>265999) &
        (_aiter5<275000 | _aiter5>275999) &
        (_aiter5<285000 | _aiter5>285999) &
        (_aiter5<295000 | _aiter5>295999) &
        (_aiter5<305000 | _aiter5>305999) &
        (_aiter5<315000 | _aiter5>315999) &
        (_aiter5<325000 | _aiter5>325999) &
        (_aiter5<335000 | _aiter5>335999) &
        (_aiter5<345000 | _aiter5>345999) &
        (_aiter5<355000 | _aiter5>355999) &
        (_aiter5<365000 | _aiter5>365999) &
        (_aiter5<375000 | _aiter5>375999) &
        (_aiter5<385000 | _aiter5>385999) &
        (_aiter5<395000 | _aiter5>395999) &
        (_aiter5<405000 | _aiter5>405999) &
        (_aiter5<415000 | _aiter5>415999) &
        (_aiter5<425000 | _aiter5>425999) &
        (_aiter5<435000 | _aiter5>435999) &
        (_aiter5<445000 | _aiter5>445999) &
        (_aiter5<455000 | _aiter5>455999) &
        (_aiter5<465000 | _aiter5>465999) &
        (_aiter5<475000 | _aiter5>475999) &
        (_aiter5<485000 | _aiter5>485999) &
        (_aiter5<495000 | _aiter5>495999) &
        (_aiter5<505000 | _aiter5>505999) &
        (_aiter5<515000 | _aiter5>515999) &
        (_aiter5<525000 | _aiter5>525999) &
        (_aiter5<535000 | _aiter5>535999) &
        (_aiter5<545000 | _aiter5>545999) &
        (_aiter5<555000 | _aiter5>555999) &
        (_aiter5<565000 | _aiter5>565999) &
        (_aiter5<575000 | _aiter5>575999) &
        (_aiter5<585000 | _aiter5>585999) &
        (_aiter5<595000 | _aiter5>595999))
```

```
      _lntotiter=_aiter5+_liter6                          ; FTYPE in order to generate total X for
      _lntotals=_lntotals+_volall[_lntotiter]              ; Lanes.

    endif
  ENDLOOP                                                 ;^End Loop 15.

  print list="\\"," ",(_lntotals/_supersuper)(10.2C)," ", PRINTO=1
  _supertotal=_supertotal+_lntotals                       ;^Generate total X for all ATYPE.

ENDLOOP                                                   ;^End Loop 14.
print list="\\"," ",(_supertotal/_supersuper)(10.2C), PRINTO=1
print list=" ","\n ", PRINTO=1

;---------------1-DIGIT FACILITY TYPES BY 1-DIGIT AREA TYPES SUMMARY----------------

Print list= "Total Summary Area Types by Facility Types ", PRINTO=1 ;^Header
Print list= "                                                    Single Digit Facility Types
", PRINTO=1
Print list= "AType         1x          2x          3x          4x          5x          6x          7x
8x          9x      Totals", PRINTO=1
Print list= "-------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1

LOOP _a1iter2=100000,599999,100000                        ;^Begin Loop 16: Cycles through ATYPE by
10 to
  _aat1=int(_a1iter2/100000)                              ; get single digit ATYPE.
  print list= _aat1(1.0),"x","    ", PRINTO=1

  _fttotal=0                                              ;^Initialize total X for all ATYPE

    LOOP _f1iter=1000,9900,1000                           ;^Begin Loop 17: Cycles through FTYPE by
10 to
                                                          ; get single digit FTYPE.
      _totftlns=0                                          ;^Initialize total X for all FTYPE by
all Lanes.
      if (_f1iter<5000 | _f1iter>5999)
      LOOP _fiter3=_f1iter,9900,100                       ;^Begin Loop 18: Cycles through two-digit
FTYPE
        if (_fiter3>_f1iter+999) BREAK                    ; for current single digit FTYPE in Loop
17.

        LOOP _aiter6=_a1iter2,599999,10000                ;^Begin Loop 19: Cycles through two-digit
ATYPE
          if (_aiter6>_a1iter2+99999) BREAK               ; for current single digit ATYPE in Loop
16.

            LOOP _liter7=1,9,1                                    ;^Begin Loop 20: Cycles through
Lanes for current FTYPE and ATYPE
              _totftlns=_totftlns+_volall[_aiter6+_fiter3+_liter7]    ; in order to generate total
X for FTYPE by ATYPE.
            ENDLOOP                                                ;^End Loop 20.

        ENDLOOP                                           ;^End Loop 19.

      ENDLOOP                                             ;^End Loop 18.
      endif
      _fttotal=_fttotal+_totftlns                         ;^Generate total X for ATYPE.

      print list="\\"," ",(_totftlns/_supersuper)(10.2C)," ", PRINTO=1
    ENDLOOP                                               ;^End Loop 17.

  print list="\\"," ",(_fttotal/_supersuper)(10.2c), PRINTO=1
ENDLOOP                                                   ;^End Loop 16.

Print list= "-------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1
print list="Totals", PRINTO=1

_supertotal=0                                             ;^Initialize overall total X.
```

```
LOOP _f1iter2=1000,9900,1000                                ;^Begin Loop 21: Cycles through FTYPE by
10
                                                            ; to get single digit FTYPE.
  _ftotals=0                                                ;^Initialize total X by FTYPE

  LOOP _fiter4=_f1iter2,9900,100                            ;^Begin Loop 22: Cycles through FTYPE by
1 to
    if (_fiter4>_f1iter2+999) BREAK                           ; get all two-digit FTYPE for current
FTYPE in
    if (_f1iter2<5000 | _f1iter2>5999)                                                    ; Loop
21.
    LOOP _liter8=1,9,1                                      ;^Begin Loop 23: Cycles through Lanes.

      LOOP _aiter7=100000,599999,10000                      ;^Begin Loop 24: Cycles through ATYPE in
order
        _ftotiter=_aiter7+_fiter4+_liter8                     ; to generate total X by single digit
FTYPE.
        _ftotals=_ftotals+_volall[_ftotiter]
      ENDLOOP                                               ;^End Loop 24.

    ENDLOOP                                                 ;^End Loop 23.
    endif
  ENDLOOP                                                   ;^End Loop 22.
  _supertotal=_supertotal+_ftotals                          ;^Generate overall total for all single
digit ATYPE
                                                            ; by all single digit FTYPE.
  print list="\\"," ",(_ftotals/_supersuper)(10.2C)," ", PRINTO=1
ENDLOOP                                                     ;^End Loop 21.

print list="\\"," ",(_supertotal/_supersuper)(10.2C), PRINTO=1
print list=" ", PRINTO=1
;*************************************************************************
; END VOLUME PERCENTAGES ON ALL LINKS REPORT
;*************************************************************************

;=========================================================================================
; BEGIN VMT ALL LINKS REPORT ------------ X = VMT ON ALL LINKS
;=========================================================================================
Print list=" ", PRINTO=1
Print
list="*****************************************************************************************
***********************************", PRINTO=1
Print                                                                                  list="*
*", PRINTO=1
Print list="*                                    VMT on All Links (Centroid Connectors Excluded)
*", PRINTO=1
Print                                                                                  list="*
*", PRINTO=1
Print
list="*****************************************************************************************
***********************************", PRINTO=1
Print list=" ", PRINTO=1
;-----------------2-DIGIT FACILITY TYPES BY 2-DIGIT AREA TYPES--------------------

LOOP _a1iter=100000,599999,100000                           ;^Begin Loop 1: Cycles through Area Types
(ATYPE) by 10
  _aat1=int(_a1iter/100000)                                  ; in order to get single digit ATYPE.
  print list= "Area Type ",_aat1(1.0),"x Range:",
           "\n ", PRINTO=1

  LOOP _aiter=_a1iter,599999,10000                           ;^Begin Loop 2: Cycles through ATYPE by
1
    if (_aiter>_a1iter+99999) BREAK                          ; in order to get two-digit ATYPE.
    _aat2=int(_aiter/10000)

    _avcheck=0                                              ;^Initialize ATYPE X checking variable.

    LOOP _achkiter=_aiter,599999,1                          ;^Begin Loop 3: Cycles through Lanes and
Facility Types (FTYPE)
      if (_achkiter>_aiter+9999) BREAK                      ; for current ATYPE in Loop 2 and totals
X checking variable.
```

```
      _avcheck=_avcheck+_vmtall[_achkiter]
    ENDLOOP                                                ;^End Loop 3.

    if (_avcheck>0)                                        ;^Begin Condition 1: If current ATYPE in
Loop 2
                                                           ; has X>0 continue to report X. Else skip
ATYPE.
      _supertotal=0                                        ;^Initialize ATYPE total X.

      Print list= "Area Type ",_aat2(2.0), PRINTO=1        ;^Header
      Print list= "                                                Number of Lanes per Direction
", PRINTO=1
      Print list= "FType          1           2           3           4           5           6
7           8           9       Totals", PRINTO=1
      Print list= "-------------------------------------------------------------------------------
-----------------------------------------------", PRINTO=1

      LOOP _fiter=100,9900,100                             ;^Begin Loop 4: Cycles through FTYPE
                                                           ; by 1 in order to get two-digit FTYPE.
       _vcheck=0                                           ;^Initialize FTYPE X checking variable.

       LOOP _liter=1,9,1                                   ;^Begin Loop 5: Cycles through Lanes for
current
         _vcheck=_vcheck+_vmtall[_aiter+_fiter+_liter]     ; FTYPE in Loop 4 and totals X checking
variable.
       ENDLOOP                                             ;^End Loop 5.

        if (_vcheck>0 & (_fiter<5000 | _fiter>5999))       ;^Begin Condition 2: If current FTYPE in
Loop 4
          _fft2=int(_fiter/100)                            ; has X>0 continue to report X. Else skip
FTYPE.
          print list= _fft2(2.0),"    ", PRINTO=1
          _totvols=0                                       ;^Initialize FTYPE total X.

          LOOP _liter2=1,9,1                               ;^Begin Loop 6: Cycles through Lanes to
generate ATYPE by FTYPE by Lanes total X.
            print list="\\"," ",_vmtall[_aiter+_fiter+_liter2](10.0C)," ", PRINTO=1
            _totvols=_totvols+_vmtall[_aiter+_fiter+_liter2]
            _supertotal=_supertotal+_vmtall[_aiter+_fiter+_liter2]
          ENDLOOP                                          ;^End Loop 6.

            print list="\\"," ",_totvols(10.0C), PRINTO=1
        endif                                              ;^End Condition 2.

      ENDLOOP                                              ;^End Loop 4.

      Print list= "-------------------------------------------------------------------------------
-----------------------------------------------", PRINTO=1
      print list="Totals", PRINTO=1

      LOOP _liter3=1,9,1                                   ;^Begin Loop 7: Cycles through Lanes for
                                                           ; current ATYPE in Loop 2.
        _lntotals=0                                        ;^Initialize Lane total X.

        LOOP _aiter2=_aiter,599999,100                     ;^Begin Loop 8: Cycles through FTYPE for
current ATYPE
          if (_aiter2>_aiter+9999) BREAK                   ; in Loop 2 to generate Lane total X.
          if (_aiter2<_aiter+5000 | _aiter2>_aiter+5999)
            _lntotiter=_aiter2+_liter3
            _lntotals=_lntotals+_vmtall[_lntotiter]
          endif
        ENDLOOP                                            ;^End Loop 8

        print list="\\"," ",_lntotals(10.0C)," ", PRINTO=1
      ENDLOOP                                              ;^End Loop 7

      print list="\\"," ",_supertotal(10.0C), PRINTO=1
      print list=" ", PRINTO=1
    endif                                                  ;^End Condition 1.

  ENDLOOP                                                  ;^End Loop 2.
```

```
  print list=" ", PRINTO=1
ENDLOOP                                                 ;^End Loop 1.

;-----------------2-DIGIT FACILITY TYPES BY TOTAL AREA TYPES--------------------

Print list= "Total Area Types ", PRINTO=1                ;^Header
Print list= "                                                    Number of Lanes per Direction
", PRINTO=1
Print list= "FType         1           2           3           4           5           6           7
8           9       Totals", PRINTO=1
Print list= "-------------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1

LOOP _fiter2=100,9900,100                               ;^Begin Loop 9: Cycles through FTYPES to
get
  _fft2=int(_fiter2/100)                                ; two-digit FTYPE.

  _tafvcheck=0                                          ;^Initialize FTYPE X checking variable.
 if (_fft2<50 | _fft2>59)
  LOOP _liter5=1,9,1                                      ;^Begin Loop 10: Cycles through Lanes
for current
                                                        ; FTYPE in Loop 9.
    LOOP _aiter4= 100000,599999,10000                     ;^Begin Loop 11: Cycles through ATYPE
for
      _tafvcheck=_tafvcheck+_vmtall[_aiter4+_fiter2+_liter5] ; current Lanes and FTYPE in order to
total X checking variable.
    ENDLOOP                                               ;^End Loop 11.

  ENDLOOP                                               ;^End Loop 10.

  if (_tafvcheck>0)                                     ;^Begin Condition 3: If current FTYPE in
Loop 9
    print list= _fft2(2.0),"    ", PRINTO=1             ; has X>0 continue to report X. Else skip
FTYPE.

    LOOP _liter4= 1,9,1                                   ;^Begin Loop 12: Cycles through Lanes
for current FTYPE
                                                        ; in Loop 9.
        _totftat=0                                      ;^Initialize FTYPE total X for all ATYPE.

      LOOP _aiter3= 100000,599999,10000                   ;^Begin Loop 13: Cycles through ATYPE
for current Lanes in Loop 12
        _totftat=_totftat+_vmtall[_aiter3+_fiter2+_liter4] ; in order to generate total X for FTYPE
by Lane for all ATYPE.
      ENDLOOP                                           ;^End Loop 13.

      print list="\\"," ",_totftat(10.0C)," ", PRINTO=1
    ENDLOOP                                             ;^End Loop 12.

    print list="\\"," ",_tafvcheck(10.0C), PRINTO=1
  endif                                                 ;^End Condition 3.
 endif
ENDLOOP                                                 ;^End Loop 9.

Print list= "-------------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1
print list="Totals", PRINTO=1

_supertotal=0                                           ;^Initialize all ATYPE total X.

LOOP _liter6=1,9,1                                      ;^Begin Loop 14: Cycles through Lanes.

  _lntotals=0                                           ;^Initialize total X for Lanes.

  LOOP _aiter5=100000,599999,100                          ;^Begin Loop 15: Cycles through ATYPE
and
    if ((_aiter5<105000 | _aiter5>105999) &
        (_aiter5<115000 | _aiter5>115999) &
        (_aiter5<125000 | _aiter5>125999) &
        (_aiter5<135000 | _aiter5>135999) &
```

```
          (_aiter5<145000 | _aiter5>145999) &
          (_aiter5<155000 | _aiter5>155999) &
          (_aiter5<165000 | _aiter5>165999) &
          (_aiter5<175000 | _aiter5>175999) &
          (_aiter5<185000 | _aiter5>185999) &
          (_aiter5<195000 | _aiter5>195999) &
          (_aiter5<205000 | _aiter5>205999) &
          (_aiter5<215000 | _aiter5>215999) &
          (_aiter5<225000 | _aiter5>225999) &
          (_aiter5<235000 | _aiter5>235999) &
          (_aiter5<245000 | _aiter5>245999) &
          (_aiter5<255000 | _aiter5>255999) &
          (_aiter5<265000 | _aiter5>265999) &
          (_aiter5<275000 | _aiter5>275999) &
          (_aiter5<285000 | _aiter5>285999) &
          (_aiter5<295000 | _aiter5>295999) &
          (_aiter5<305000 | _aiter5>305999) &
          (_aiter5<315000 | _aiter5>315999) &
          (_aiter5<325000 | _aiter5>325999) &
          (_aiter5<335000 | _aiter5>335999) &
          (_aiter5<345000 | _aiter5>345999) &
          (_aiter5<355000 | _aiter5>355999) &
          (_aiter5<365000 | _aiter5>365999) &
          (_aiter5<375000 | _aiter5>375999) &
          (_aiter5<385000 | _aiter5>385999) &
          (_aiter5<395000 | _aiter5>395999) &
          (_aiter5<405000 | _aiter5>405999) &
          (_aiter5<415000 | _aiter5>415999) &
          (_aiter5<425000 | _aiter5>425999) &
          (_aiter5<435000 | _aiter5>435999) &
          (_aiter5<445000 | _aiter5>445999) &
          (_aiter5<455000 | _aiter5>455999) &
          (_aiter5<465000 | _aiter5>465999) &
          (_aiter5<475000 | _aiter5>475999) &
          (_aiter5<485000 | _aiter5>485999) &
          (_aiter5<495000 | _aiter5>495999) &
          (_aiter5<505000 | _aiter5>505999) &
          (_aiter5<515000 | _aiter5>515999) &
          (_aiter5<525000 | _aiter5>525999) &
          (_aiter5<535000 | _aiter5>535999) &
          (_aiter5<545000 | _aiter5>545999) &
          (_aiter5<555000 | _aiter5>555999) &
          (_aiter5<565000 | _aiter5>565999) &
          (_aiter5<575000 | _aiter5>575999) &
          (_aiter5<585000 | _aiter5>585999) &
          (_aiter5<595000 | _aiter5>595999))

       _lntotiter=_aiter5+_liter6                          ; FTYPE in order to generate total X for
       _lntotals=_lntotals+_vmtall[_lntotiter]              ; Lanes.

     endif
   ENDLOOP                                                ;^End Loop 15.

   print list="\\"," ",_lntotals(10.0C)," ", PRINTO=1
   _supertotal=_supertotal+_lntotals                      ;^Generate total X for all ATYPE.

ENDLOOP                                                   ;^End Loop 14.
print list="\\"," ",_supertotal(10.0C), PRINTO=1
print list=" ","\n ", PRINTO=1

;---------------1-DIGIT FACILITY TYPES BY 1-DIGIT AREA TYPES SUMMARY----------------

Print list= "Total Summary Area Types by Facility Types ", PRINTO=1 ;^Header
Print list= "                                                        Single Digit Facility Types
", PRINTO=1
Print list= "AType         1x          2x          3x          4x          5x          6x          7x
8x          9x      Totals", PRINTO=1
Print list= "-------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1
```

```
LOOP _a1iter2=100000,599999,100000                          ;^Begin Loop 16: Cycles through ATYPE by
10 to
  _aat1=int(_a1iter2/100000)                                 ; get single digit ATYPE.
  print list= _aat1(1.0),"x","    ", PRINTO=1

  _fttotal=0                                                 ;^Initialize total X for all ATYPE

    LOOP _f1iter=1000,9900,1000                              ;^Begin Loop 17: Cycles through FTYPE by
10 to
                                                             ; get single digit FTYPE.
      _totftlns=0                                              ;^Initialize total X for all FTYPE by
all Lanes.
      if (_f1iter<5000 | _f1iter>5999)
      LOOP _fiter3=_f1iter,9900,100                          ;^Begin Loop 18: Cycles through two-digit
FTYPE
        if (_fiter3>_f1iter+999) BREAK                       ; for current single digit FTYPE in Loop
17.

        LOOP _aiter6=_a1iter2,599999,10000                   ;^Begin Loop 19: Cycles through two-digit
ATYPE
          if (_aiter6>_a1iter2+99999) BREAK                  ; for current single digit ATYPE in Loop
16.

            LOOP _liter7=1,9,1                                       ;^Begin Loop 20: Cycles through
Lanes for current FTYPE and ATYPE
              _totftlns=_totftlns+_vmtall[_aiter6+_fiter3+_liter7]   ; in order to generate total
X for FTYPE by ATYPE.
            ENDLOOP                                                   ;^End Loop 20.

        ENDLOOP                                              ;^End Loop 19.


      ENDLOOP                                                ;^End Loop 18.
      endif
      _fttotal=_fttotal+_totftlns                            ;^Generate total X for ATYPE.

      print list="\\"," ",_totftlns(10.0C)," ", PRINTO=1
    ENDLOOP                                                  ;^End Loop 17.

  print list="\\"," ",_fttotal(10.0c), PRINTO=1
ENDLOOP                                                      ;^End Loop 16.

Print list= "-------------------------------------------------------------------------------------
-----------------------------------------", PRINTO=1
print list="Totals", PRINTO=1

_supertotal=0                                                ;^Initialize overall total X.

LOOP _f1iter2=1000,9900,1000                                 ;^Begin Loop 21: Cycles through FTYPE by
10
                                                             ; to get single digit FTYPE.
  _ftotals=0                                                 ;^Initialize total X by FTYPE

  LOOP _fiter4=_f1iter2,9900,100                             ;^Begin Loop 22: Cycles through FTYPE by
1 to
    if (_fiter4>_f1iter2+999) BREAK                            ; get all two-digit FTYPE for current
FTYPE in
    if (_f1iter2<5000 | _f1iter2>5999)                                                    ; Loop
21.
    LOOP _liter8=1,9,1                                       ;^Begin Loop 23: Cycles through Lanes.

      LOOP _aiter7=100000,599999,10000                       ;^Begin Loop 24: Cycles through ATYPE in
order
        _ftotiter=_aiter7+_fiter4+_liter8                      ; to generate total X by single digit
FTYPE.
        _ftotals=_ftotals+_vmtall[_ftotiter]
      ENDLOOP                                                ;^End Loop 24.

    ENDLOOP                                                  ;^End Loop 23.
    endif
  ENDLOOP                                                    ;^End Loop 22.
```

```
  _supertotal=_supertotal+_ftotals                          ;^Generate overall total for all single
digit ATYPE
                                                            ; by all single digit FTYPE.
  print list="\\"," ",_ftotals(10.0C)," ", PRINTO=1
ENDLOOP                                                     ;^End Loop 21.
_totalvmt=_supertotal
print list="\\"," ",_supertotal(10.0C), PRINTO=1
print list=" ", PRINTO=1
;*********************************************************************
; END VMT ALL LINKS REPORT
;*********************************************************************

;=============================================================================================
; BEGIN VHT ALL LINKS REPORT ------------ X = VHT ON ALL LINKS
;=============================================================================================
Print list=" ", PRINTO=1
Print
list="*************************************************************************************************
***********************************", PRINTO=1
Print                                                                                          list="*
*", PRINTO=1
Print list="*                                        VHT on All Links (Centroid Connectors Excluded)
*", PRINTO=1
Print                                                                                          list="*
*", PRINTO=1
Print
list="*************************************************************************************************
***********************************", PRINTO=1
Print list=" ", PRINTO=1
;-----------------2-DIGIT FACILITY TYPES BY 2-DIGIT AREA TYPES--------------------

LOOP _a1iter=100000,599999,100000                          ;^Begin Loop 1: Cycles through Area Types
(ATYPE) by 10
  _aat1=int(_a1iter/100000)                                 ; in order to get single digit ATYPE.
  print list= "Area Type ",_aat1(1.0),"x Range:",
           "\n ", PRINTO=1

  LOOP _aiter=_a1iter,599999,10000                          ;^Begin Loop 2: Cycles through ATYPE by
1
    if (_aiter>_a1iter+99999) BREAK                         ; in order to get two-digit ATYPE.
    _aat2=int(_aiter/10000)

    _avcheck=0                                              ;^Initialize ATYPE X checking variable.

    LOOP _achkiter=_aiter,599999,1                         ;^Begin Loop 3: Cycles through Lanes and
Facility Types (FTYPE)
      if (_achkiter>_aiter+9999) BREAK                      ; for current ATYPE in Loop 2 and totals
X checking variable.
      _avcheck=_avcheck+_vhtall[_achkiter]
    ENDLOOP                                                 ;^End Loop 3.

    if (_avcheck>0)                                        ;^Begin Condition 1: If current ATYPE in
Loop 2
                                                            ; has X>0 continue to report X. Else skip
ATYPE.
      _supertotal=0                                         ;^Initialize ATYPE total X.

      Print list= "Area Type ",_aat2(2.0), PRINTO=1         ;^Header
      Print list= "                                                    Number of Lanes per Direction
", PRINTO=1
      Print list= "FType          1           2           3           4           5           6
7           8           9       Totals", PRINTO=1
      Print list= "-------------------------------------------------------------------------------
-----------------------------------------------", PRINTO=1

      LOOP _fiter=100,9900,100                              ;^Begin Loop 4: Cycles through FTYPE
                                                            ; by 1 in order to get two-digit FTYPE.
       _vcheck=0                                            ;^Initialize FTYPE X checking variable.

       LOOP _liter=1,9,1                                   ;^Begin Loop 5: Cycles through Lanes for
current
```

```
          _vcheck=_vcheck+_vhtall[_aiter+_fiter+_liter]    ; FTYPE in Loop 4 and totals X checking
variable.
       ENDLOOP                                            ;^End Loop 5.

        if (_vcheck>0 & (_fiter<5000 | _fiter>5999))      ;^Begin Condition 2: If current FTYPE in
Loop 4
          _fft2=int(_fiter/100)                           ; has X>0 continue to report X. Else skip
FTYPE.
          print list= _fft2(2.0),"    ", PRINTO=1
          _totvols=0                                      ;^Initialize FTYPE total X.

          LOOP _liter2=1,9,1                               ;^Begin Loop 6: Cycles through Lanes to
generate ATYPE by FTYPE by Lanes total X.
            print list="\\"," ",_vhtall[_aiter+_fiter+_liter2](10.0C)," ", PRINTO=1
            _totvols=_totvols+_vhtall[_aiter+_fiter+_liter2]
            _supertotal=_supertotal+_vhtall[_aiter+_fiter+_liter2]
          ENDLOOP                                         ;^End Loop 6.

            print list="\\"," ",_totvols(10.0C), PRINTO=1
        endif                                             ;^End Condition 2.

      ENDLOOP                                             ;^End Loop 4.

      Print list= "-------------------------------------------------------------------------------
-----------------------------------------------", PRINTO=1
      print list="Totals", PRINTO=1

      LOOP _liter3=1,9,1                                 ;^Begin Loop 7: Cycles through Lanes for
                                                         ; current ATYPE in Loop 2.
        _lntotals=0                                      ;^Initialize Lane total X.

        LOOP _aiter2=_aiter,599999,100                   ;^Begin Loop 8: Cycles through FTYPE for
current ATYPE
          if (_aiter2>_aiter+9999) BREAK                  ; in Loop 2 to generate Lane total X.
          if (_aiter2<_aiter+5000 | _aiter2>_aiter+5999)
            _lntotiter=_aiter2+_liter3
            _lntotals=_lntotals+_vhtall[_lntotiter]
          endif
        ENDLOOP                                          ;^End Loop 8

        print list="\\"," ",_lntotals(10.0C)," ", PRINTO=1
      ENDLOOP                                            ;^End Loop 7

      print list="\\"," ",_supertotal(10.0C), PRINTO=1
      print list=" ", PRINTO=1
    endif                                                ;^End Condition 1.

  ENDLOOP                                                ;^End Loop 2.

  print list=" ", PRINTO=1
ENDLOOP                                                  ;^End Loop 1.

;-----------------2-DIGIT FACILITY TYPES BY TOTAL AREA TYPES--------------------

Print list= "Total Area Types ", PRINTO=1                ;^Header
Print list= "                                                    Number of Lanes per Direction
", PRINTO=1
Print list= "FType         1          2          3          4          5          6          7
8          9       Totals", PRINTO=1
Print list= "-------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1

LOOP _fiter2=100,9900,100                                ;^Begin Loop 9: Cycles through FTYPES to
get
  _fft2=int(_fiter2/100)                                 ; two-digit FTYPE.

  _tafvcheck=0                                           ;^Initialize FTYPE X checking variable.
 if (_fft2<50 | _fft2>59)
  LOOP _liter5=1,9,1                                       ;^Begin Loop 10: Cycles through Lanes
for current
                                                         ; FTYPE in Loop 9.
```

```
    LOOP _aiter4= 100000,599999,10000                          ;^Begin Loop 11: Cycles through ATYPE
for
      _tafvcheck=_tafvcheck+_vhtall[_aiter4+_fiter2+_liter5] ; current Lanes and FTYPE in order to
total X checking variable.

    ENDLOOP                                                    ;^End Loop 11.

  ENDLOOP                                                     ;^End Loop 10.

  if (_tafvcheck>0)                                           ;^Begin Condition 3: If current FTYPE in
Loop 9
    print list= _fft2(2.0),"    ", PRINTO=1                   ; has X>0 continue to report X. Else skip
FTYPE.

    LOOP _liter4= 1,9,1                                        ;^Begin Loop 12: Cycles through Lanes
for current FTYPE
                                                               ; in Loop 9.
        _totftat=0                                            ;^Initialize FTYPE total X for all ATYPE.

      LOOP _aiter3= 100000,599999,10000                        ;^Begin Loop 13: Cycles through ATYPE
for current Lanes in Loop 12
        _totftat=_totftat+_vhtall[_aiter3+_fiter2+_liter4] ; in order to generate total X for FTYPE
by Lane for all ATYPE.
      ENDLOOP                                                 ;^End Loop 13.

      print list="\\"," ",_totftat(10.0C)," ", PRINTO=1
    ENDLOOP                                                   ;^End Loop 12.

    print list="\\"," ",_tafvcheck(10.0C), PRINTO=1
  endif                                                       ;^End Condition 3.
 endif
ENDLOOP                                                       ;^End Loop 9.

Print list= "-----------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1
print list="Totals", PRINTO=1

_supertotal=0                                                 ;^Initialize all ATYPE total X.

LOOP _liter6=1,9,1                                            ;^Begin Loop 14: Cycles through Lanes.

  _lntotals=0                                                 ;^Initialize total X for Lanes.

  LOOP _aiter5=100000,599999,100                               ;^Begin Loop 15: Cycles through ATYPE
and
    if ((_aiter5<105000 | _aiter5>105999) &
        (_aiter5<115000 | _aiter5>115999) &
        (_aiter5<125000 | _aiter5>125999) &
        (_aiter5<135000 | _aiter5>135999) &
        (_aiter5<145000 | _aiter5>145999) &
        (_aiter5<155000 | _aiter5>155999) &
        (_aiter5<165000 | _aiter5>165999) &
        (_aiter5<175000 | _aiter5>175999) &
        (_aiter5<185000 | _aiter5>185999) &
        (_aiter5<195000 | _aiter5>195999) &
        (_aiter5<205000 | _aiter5>205999) &
        (_aiter5<215000 | _aiter5>215999) &
        (_aiter5<225000 | _aiter5>225999) &
        (_aiter5<235000 | _aiter5>235999) &
        (_aiter5<245000 | _aiter5>245999) &
        (_aiter5<255000 | _aiter5>255999) &
        (_aiter5<265000 | _aiter5>265999) &
        (_aiter5<275000 | _aiter5>275999) &
        (_aiter5<285000 | _aiter5>285999) &
        (_aiter5<295000 | _aiter5>295999) &
        (_aiter5<305000 | _aiter5>305999) &
        (_aiter5<315000 | _aiter5>315999) &
        (_aiter5<325000 | _aiter5>325999) &
        (_aiter5<335000 | _aiter5>335999) &
        (_aiter5<345000 | _aiter5>345999) &
        (_aiter5<355000 | _aiter5>355999) &
```

```
        (_aiter5<365000 | _aiter5>365999) &
        (_aiter5<375000 | _aiter5>375999) &
        (_aiter5<385000 | _aiter5>385999) &
        (_aiter5<395000 | _aiter5>395999) &
        (_aiter5<405000 | _aiter5>405999) &
        (_aiter5<415000 | _aiter5>415999) &
        (_aiter5<425000 | _aiter5>425999) &
        (_aiter5<435000 | _aiter5>435999) &
        (_aiter5<445000 | _aiter5>445999) &
        (_aiter5<455000 | _aiter5>455999) &
        (_aiter5<465000 | _aiter5>465999) &
        (_aiter5<475000 | _aiter5>475999) &
        (_aiter5<485000 | _aiter5>485999) &
        (_aiter5<495000 | _aiter5>495999) &
        (_aiter5<505000 | _aiter5>505999) &
        (_aiter5<515000 | _aiter5>515999) &
        (_aiter5<525000 | _aiter5>525999) &
        (_aiter5<535000 | _aiter5>535999) &
        (_aiter5<545000 | _aiter5>545999) &
        (_aiter5<555000 | _aiter5>555999) &
        (_aiter5<565000 | _aiter5>565999) &
        (_aiter5<575000 | _aiter5>575999) &
        (_aiter5<585000 | _aiter5>585999) &
        (_aiter5<595000 | _aiter5>595999))

      _lntotiter=_aiter5+_liter6                          ; FTYPE in order to generate total X for
      _lntotals=_lntotals+_vhtall[_lntotiter]              ; Lanes.

    endif
  ENDLOOP                                                 ;^End Loop 15.

  print list="\\"," ",_lntotals(10.0C)," ", PRINTO=1
  _supertotal=_supertotal+_lntotals                       ;^Generate total X for all ATYPE.

ENDLOOP                                                   ;^End Loop 14.
print list="\\"," ",_supertotal(10.0C), PRINTO=1
print list=" ","\n ", PRINTO=1

;---------------1-DIGIT FACILITY TYPES BY 1-DIGIT AREA TYPES SUMMARY----------------

Print list= "Total Summary Area Types by Facility Types ", PRINTO=1 ;^Header
Print list= "                                                        Single Digit Facility Types
", PRINTO=1
Print list= "AType         1x          2x          3x          4x          5x          6x          7x
8x          9x      Totals", PRINTO=1
Print list= "-------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1

LOOP _a1iter2=100000,599999,100000                        ;^Begin Loop 16: Cycles through ATYPE by
10 to
  _aat1=int(_a1iter2/100000)                              ; get single digit ATYPE.
  print list= _aat1(1.0),"x","    ", PRINTO=1

  _fttotal=0                                              ;^Initialize total X for all ATYPE

    LOOP _f1iter=1000,9900,1000                           ;^Begin Loop 17: Cycles through FTYPE by
10 to
                                                          ; get single digit FTYPE.
      _totftlns=0                                           ;^Initialize total X for all FTYPE by
all Lanes.
      if (_f1iter<5000 | _f1iter>5999)
      LOOP _fiter3=_f1iter,9900,100                       ;^Begin Loop 18: Cycles through two-digit
FTYPE
        if (_fiter3>_f1iter+999) BREAK                    ; for current single digit FTYPE in Loop
17.

        LOOP _aiter6=_a1iter2,599999,10000                ;^Begin Loop 19: Cycles through two-digit
ATYPE
          if (_aiter6>_a1iter2+99999) BREAK               ; for current single digit ATYPE in Loop
16.
```

```
            LOOP _liter7=1,9,1                                          ;^Begin Loop 20: Cycles through
Lanes for current FTYPE and ATYPE
              _totftlns=_totftlns+_vhtall[_aiter6+_fiter3+_liter7]    ; in order to generate total
X for FTYPE by ATYPE.
            ENDLOOP                                                      ;^End Loop 20.

        ENDLOOP                                                 ;^End Loop 19.

      ENDLOOP                                                   ;^End Loop 18.
      endif
      _fttotal=_fttotal+_totftlns                               ;^Generate total X for ATYPE.

      print list="\\"," ",_totftlns(10.0C)," ", PRINTO=1
    ENDLOOP                                                     ;^End Loop 17.

  print list="\\"," ",_fttotal(10.0c), PRINTO=1
ENDLOOP                                                         ;^End Loop 16.

Print list= "-------------------------------------------------------------------------------------
-----------------------------------------", PRINTO=1
print list="Totals", PRINTO=1

_supertotal=0                                                   ;^Initialize overall total X.

LOOP _f1iter2=1000,9900,1000                                    ;^Begin Loop 21: Cycles through FTYPE by
10
                                                                ; to get single digit FTYPE.
  _ftotals=0                                                    ;^Initialize total X by FTYPE

  LOOP _fiter4=_f1iter2,9900,100                                ;^Begin Loop 22: Cycles through FTYPE by
1 to
    if (_fiter4>_f1iter2+999) BREAK                               ; get all two-digit FTYPE for current
FTYPE in
    if (_f1iter2<5000 | _f1iter2>5999)                                                          ; Loop
21.
    LOOP _liter8=1,9,1                                          ;^Begin Loop 23: Cycles through Lanes.

      LOOP _aiter7=100000,599999,10000                          ;^Begin Loop 24: Cycles through ATYPE in
order
        _ftotiter=_aiter7+_fiter4+_liter8                         ; to generate total X by single digit
FTYPE.
        _ftotals=_ftotals+_vhtall[_ftotiter]
      ENDLOOP                                                   ;^End Loop 24.

    ENDLOOP                                                     ;^End Loop 23.
    endif
  ENDLOOP                                                       ;^End Loop 22.
  _supertotal=_supertotal+_ftotals                              ;^Generate overall total for all single
digit ATYPE
                                                                ; by all single digit FTYPE.
  print list="\\"," ",_ftotals(10.0C)," ", PRINTO=1
ENDLOOP                                                         ;^End Loop 21.
_totalvht=_supertotal
print list="\\"," ",_supertotal(10.0C), PRINTO=1
print list=" ", PRINTO=1
;*********************************************************************
; END VHT ALL LINKS REPORT
;*********************************************************************

;=============================================================================================
; BEGIN FREE FLOW SPEED REPORT ------------- X = Free Flow Speeds
;=============================================================================================
Print list=" ", PRINTO=1
Print
list="*********************************************************************************************
*************************************", PRINTO=1
Print                                                                                    list="*
*", PRINTO=1
Print list="*                                                          Original Speed (MPH)
*", PRINTO=1
```

```
Print                                                                                    list="*
*", PRINTO=1
Print
list="*************************************************************************************
**********************************", PRINTO=1
Print list=" ", PRINTO=1
;-----------------2-DIGIT FACILITY TYPES BY 2-DIGIT AREA TYPES--------------------
LOOP _a1iter=100000,599999,100000                          ;^Begin Loop 1: Cycles through Area Types
(ATYPE) by 10
  _aat1=int(_a1iter/100000)                                 ; in order to get single digit ATYPE.
  print list= "Area Type ",_aat1(1.0),"x Range:",
           "\n ", PRINTO=1

  LOOP _aiter=_a1iter,599999,10000                          ;^Begin Loop 2: Cycles through ATYPE by
1
    if (_aiter>_a1iter+99999) BREAK                         ; in order to get two-digit ATYPE.
    _aat2=int(_aiter/10000)

    _avcheck=0                                              ;^Initialize ATYPE X checking variable.
    _avdist=0

    LOOP _achkiter=_aiter,599999,1                         ;^Begin Loop 3: Cycles through Lanes and
Facility Types (FTYPE)
      if (_achkiter>_aiter+9999) BREAK                      ; for current ATYPE in Loop 2 and totals
X checking variable.
      _avcheck=_avcheck+_wffspd[_achkiter]
    ENDLOOP                                                 ;^End Loop 3.

    if (_avcheck>0)                                        ;^Begin Condition 1: If current ATYPE in
Loop 2
                                                            ; has X>0 continue to report X. Else skip
ATYPE.
      _supertotal=0                                         ;^Initialize ATYPE total X.
      _superdist=0

      Print list= "Area Type ",_aat2(2.0), PRINTO=1        ;^Header
      Print list= "                                                  Number of Lanes per Direction
", PRINTO=1
      Print list= "FType          1           2           3           4           5           6
7           8           9       Totals", PRINTO=1
      Print list= "-------------------------------------------------------------------------------
-----------------------------------------------", PRINTO=1

      LOOP _fiter=100,9900,100                              ;^Begin Loop 4: Cycles through FTYPE
                                                            ; by 1 in order to get two-digit FTYPE.
       _vcheck=0                                            ;^Initialize FTYPE X checking variable.
       _vdist=0

       LOOP _liter=1,9,1                                   ;^Begin Loop 5: Cycles through Lanes for
current
         _vcheck=_vcheck+_wffspd[_aiter+_fiter+_liter]     ; FTYPE in Loop 4 and totals X checking
variable.
       ENDLOOP                                              ;^End Loop 5.

        if (_vcheck>0 & (_fiter<5000 | _fiter>5999))       ;^Begin Condition 2: If current FTYPE in
Loop 4
          _fft2=int(_fiter/100)                            ; has X>0 continue to report X. Else skip
FTYPE.
          print list= _fft2(2.0),"    ", PRINTO=1
          _totvols=0                                        ;^Initialize FTYPE total X.
          _totdist=0

          LOOP _liter2=1,9,1                                ;^Begin Loop 6: Cycles through Lanes to
generate ATYPE by FTYPE by Lanes total X.
            if (_dmiles[_aiter+_fiter+_liter2]>0)
              _spdspd=_wffspd[_aiter+_fiter+_liter2]/_dmiles[_aiter+_fiter+_liter2]
            else
              _spdspd=0
            endif
            print list="\\"," ",_spdspd(10.2C)," ", PRINTO=1
            _totvols=_totvols+_wffspd[_aiter+_fiter+_liter2]
```

```
            _totdist=_totdist+_dmiles[_aiter+_fiter+_liter2]
            _supertotal=_supertotal+_wffspd[_aiter+_fiter+_liter2]
            _superdist=_superdist+_dmiles[_aiter+_fiter+_liter2]

          ENDLOOP                                          ;^End Loop 6
            if (_totdist>0)
              _totspd=_totvols/_totdist
            else
              _totspd=0
            endif
            print list="\\"," ",_totspd(10.2C), PRINTO=1
        endif                                              ;^End Condition 2.

      ENDLOOP                                              ;^End Loop 4.


      Print list= "-------------------------------------------------------------------------------
-----------------------------------------------", PRINTO=1
      print list="Totals", PRINTO=1

      LOOP _liter3=1,9,1                                  ;^Begin Loop 7: Cycles through Lanes for
                                                          ; current ATYPE in Loop 2.
        _lntotals=0                                       ;^Initialize Lane total X.
        _lndist=0

        LOOP _aiter2=_aiter,599999,100                    ;^Begin Loop 8: Cycles through FTYPE for
current ATYPE
          if (_aiter2>_aiter+9999) BREAK                  ; in Loop 2 to generate Lane total X.
          if (_aiter2<_aiter+5000 | _aiter2>_aiter+5999)
            _lntotiter=_aiter2+_liter3
            _lntotals=_lntotals+_wffspd[_lntotiter]
            _lndist=_lndist+_dmiles[_lntotiter]
          endif
        ENDLOOP                                           ;^End Loop 8

        if (_lndist>0)
          _lnspd=_lntotals/_lndist
        else
          _lnspd=0
        endif
        print list="\\"," ",_lnspd(10.2C)," ", PRINTO=1
      ENDLOOP                                             ;^End Loop 7

      if (_superdist>0)
        _superspd=_supertotal/_superdist
      else
        _superspd=0
      endif
      print list="\\"," ",_superspd(10.2C), PRINTO=1
      print list=" ", PRINTO=1
    endif                                                 ;^End Condition 1.

  ENDLOOP                                                 ;^End Loop 2.

  print list=" ", PRINTO=1
ENDLOOP                                                   ;^End Loop 1.

;-----------------2-DIGIT FACILITY TYPES BY TOTAL AREA TYPES--------------------

Print list= "Total Area Types ", PRINTO=1                 ;^Header
Print list= "                                                      Number of Lanes per Direction
", PRINTO=1
Print list= "FType         1           2           3           4           5           6           7
8           9       Totals", PRINTO=1
Print list= "-------------------------------------------------------------------------------------
-----------------------------------------", PRINTO=1

LOOP _fiter2=100,9900,100                                 ;^Begin Loop 9: Cycles through FTYPES to
get
  _fft2=int(_fiter2/100)                                  ; two-digit FTYPE.
```

```
  _tafvcheck=0                                                ;^Initialize FTYPE X checking variable.
  _tafvdist=0

 if (_fft2<50 | _fft2>59)
  LOOP _liter5=1,9,1                                            ;^Begin Loop 10: Cycles through Lanes
for current
                                                              ; FTYPE in Loop 9.
    LOOP _aiter4= 100000,599999,10000                           ;^Begin Loop 11: Cycles through ATYPE
for
      _tafvcheck=_tafvcheck+_wffspd[_aiter4+_fiter2+_liter5] ; current Lanes and FTYPE in order to
total X checking variable.
      _tafvdist=_tafvdist+_dmiles[_aiter4+_fiter2+_liter5]
    ENDLOOP                                                     ;^End Loop 11.

  ENDLOOP                                                     ;^End Loop 10.

  if (_tafvcheck>0)                                           ;^Begin Condition 3: If current FTYPE in
Loop 9
    print list= _fft2(2.0),"    ", PRINTO=1                   ; has X>0 continue to report X. Else skip
FTYPE.

    LOOP _liter4= 1,9,1                                         ;^Begin Loop 12: Cycles through Lanes
for current FTYPE
                                                              ; in Loop 9.
        _totftat=0                                           ;^Initialize FTYPE total X for all ATYPE.
        _totftatdist=0

      LOOP _aiter3= 100000,599999,10000                         ;^Begin Loop 13: Cycles through ATYPE
for current Lanes in Loop 12
        _totftat=_totftat+_wffspd[_aiter3+_fiter2+_liter4] ; in order to generate total X for FTYPE
by Lane for all ATYPE.
        _totftatdist=_totftatdist+_dmiles[_aiter3+_fiter2+_liter4]
      ENDLOOP                                                 ;^End Loop 13.

      if (_totftatdist>0)
        _totftatspd=_totftat/_totftatdist
      else
        _totftatspd=0
      endif
      print list="\\"," ",_totftatspd(10.2C)," ", PRINTO=1
    ENDLOOP                                                   ;^End Loop 12.

    if (_tafvdist>0)
      _tafvspd=_tafvcheck/_tafvdist
    else
      _tafvspd=0
    endif
    print list="\\"," ",_tafvspd(10.2C), PRINTO=1
  endif                                                       ;^End Condition 3.
 endif
ENDLOOP                                                       ;^End Loop 9.

Print list= "-------------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1
print list="Totals", PRINTO=1

_supertotal=0                                                 ;^Initialize all ATYPE total X.
_superdist=0

LOOP _liter6=1,9,1                                            ;^Begin Loop 14: Cycles through Lanes.

  _lntotals=0                                                 ;^Initialize total X for Lanes.
  _lndist=0

  LOOP _aiter5=100000,599999,100                                ;^Begin Loop 15: Cycles through ATYPE
and
    if ((_aiter5<105000 | _aiter5>105999) &
        (_aiter5<115000 | _aiter5>115999) &
        (_aiter5<125000 | _aiter5>125999) &
        (_aiter5<135000 | _aiter5>135999) &
        (_aiter5<145000 | _aiter5>145999) &
```

```
          (_aiter5<155000 | _aiter5>155999) &
          (_aiter5<165000 | _aiter5>165999) &
          (_aiter5<175000 | _aiter5>175999) &
          (_aiter5<185000 | _aiter5>185999) &
          (_aiter5<195000 | _aiter5>195999) &
          (_aiter5<205000 | _aiter5>205999) &
          (_aiter5<215000 | _aiter5>215999) &
          (_aiter5<225000 | _aiter5>225999) &
          (_aiter5<235000 | _aiter5>235999) &
          (_aiter5<245000 | _aiter5>245999) &
          (_aiter5<255000 | _aiter5>255999) &
          (_aiter5<265000 | _aiter5>265999) &
          (_aiter5<275000 | _aiter5>275999) &
          (_aiter5<285000 | _aiter5>285999) &
          (_aiter5<295000 | _aiter5>295999) &
          (_aiter5<305000 | _aiter5>305999) &
          (_aiter5<315000 | _aiter5>315999) &
          (_aiter5<325000 | _aiter5>325999) &
          (_aiter5<335000 | _aiter5>335999) &
          (_aiter5<345000 | _aiter5>345999) &
          (_aiter5<355000 | _aiter5>355999) &
          (_aiter5<365000 | _aiter5>365999) &
          (_aiter5<375000 | _aiter5>375999) &
          (_aiter5<385000 | _aiter5>385999) &
          (_aiter5<395000 | _aiter5>395999) &
          (_aiter5<405000 | _aiter5>405999) &
          (_aiter5<415000 | _aiter5>415999) &
          (_aiter5<425000 | _aiter5>425999) &
          (_aiter5<435000 | _aiter5>435999) &
          (_aiter5<445000 | _aiter5>445999) &
          (_aiter5<455000 | _aiter5>455999) &
          (_aiter5<465000 | _aiter5>465999) &
          (_aiter5<475000 | _aiter5>475999) &
          (_aiter5<485000 | _aiter5>485999) &
          (_aiter5<495000 | _aiter5>495999) &
          (_aiter5<505000 | _aiter5>505999) &
          (_aiter5<515000 | _aiter5>515999) &
          (_aiter5<525000 | _aiter5>525999) &
          (_aiter5<535000 | _aiter5>535999) &
          (_aiter5<545000 | _aiter5>545999) &
          (_aiter5<555000 | _aiter5>555999) &
          (_aiter5<565000 | _aiter5>565999) &
          (_aiter5<575000 | _aiter5>575999) &
          (_aiter5<585000 | _aiter5>585999) &
          (_aiter5<595000 | _aiter5>595999))

        _lntotiter=_aiter5+_liter6                          ; FTYPE in order to generate total X for
        _lntotals=_lntotals+_wffspd[_lntotiter]              ; Lanes.
        _lndist=_lndist+_dmiles[_lntotiter]

      endif
    ENDLOOP                                                 ;^End Loop 15.

    if (_lndist>0)
      _lnspd=_lntotals/_lndist
    else
      _lnspd=0
    endif
    print list="\\"," ",_lnspd(10.2C)," ", PRINTO=1
    _supertotal=_supertotal+_lntotals                       ;^Generate total X for all ATYPE.
    _superdist=_superdist+_lndist

  ENDLOOP                                                   ;^End Loop 14.
  if (_superdist>0)
    _superspd=_supertotal/_superdist
  else
    _superspd=0
  endif
  print list="\\"," ",_superspd(10.2C), PRINTO=1
  print list=" ","\n ", PRINTO=1
```

```
;---------------1-DIGIT FACILITY TYPES BY 1-DIGIT AREA TYPES SUMMARY----------------

Print list= "Total Summary Area Types by Facility Types ", PRINTO=1 ;^Header
Print list= "                                                    Single Digit Facility Types
", PRINTO=1
Print list= "AType         1x          2x          3x          4x          5x          6x          7x
8x          9x      Totals", PRINTO=1
Print list= "-------------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1

LOOP _a1iter2=100000,599999,100000                          ;^Begin Loop 16: Cycles through ATYPE by
10 to
  _aat1=int(_a1iter2/100000)                                ; get single digit ATYPE.
  print list= _aat1(1.0),"x","    ", PRINTO=1

  _fttotal=0                                                ;^Initialize total X for all ATYPE
  _ftdist=0

    LOOP _f1iter=1000,9900,1000                             ;^Begin Loop 17: Cycles through FTYPE by
10 to
                                                            ; get single digit FTYPE.
      _totftlns=0                                             ;^Initialize total X for all FTYPE by
all Lanes.
      _totftlnsdist=0
      if (_f1iter<5000 | _f1iter>5999)
      LOOP _fiter3=_f1iter,9900,100                         ;^Begin Loop 18: Cycles through two-digit
FTYPE
        if (_fiter3>_f1iter+999) BREAK                      ; for current single digit FTYPE in Loop
17.

        LOOP _aiter6=_a1iter2,599999,10000                  ;^Begin Loop 19: Cycles through two-digit
ATYPE
          if (_aiter6>_a1iter2+99999) BREAK                 ; for current single digit ATYPE in Loop
16.

            LOOP _liter7=1,9,1                                      ;^Begin Loop 20: Cycles through
Lanes for current FTYPE and ATYPE
              _totftlns=_totftlns+_wffspd[_aiter6+_fiter3+_liter7]   ; in order to generate total
X for FTYPE by ATYPE.
              _totftlnsdist=_totftlnsdist+_dmiles[_aiter6+_fiter3+_liter7]
            ENDLOOP                                                  ;^End Loop 20.

        ENDLOOP                                             ;^End Loop 19.

      ENDLOOP                                               ;^End Loop 18.
      endif
      _fttotal=_fttotal+_totftlns                           ;^Generate total X for ATYPE.
      _ftdist=_ftdist+_totftlnsdist

      if (_totftlnsdist>0)
        _totftlnsspd=_totftlns/_totftlnsdist
      else
        _totftlnsspd=0
      endif
      print list="\\"," ",_totftlnsspd(10.2C)," ", PRINTO=1
    ENDLOOP                                                 ;^End Loop 17.

  if (_ftdist>0)
    _ftspd=_fttotal/_ftdist
  else
    _ftspd=0
  endif
  print list="\\"," ",_ftspd(10.2c), PRINTO=1
ENDLOOP                                                     ;^End Loop 16.

Print list= "-------------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1
print list="Totals", PRINTO=1

_supertotal=0                                               ;^Initialize overall total X.
_superdist=0
```

```
LOOP _f1iter2=1000,9900,1000                              ;^Begin Loop 21: Cycles through FTYPE by
10
                                                           ; to get single digit FTYPE.
  _ftotals=0                                              ;^Initialize total X by FTYPE
  _ftdist=0

  LOOP _fiter4=_f1iter2,9900,100                          ;^Begin Loop 22: Cycles through FTYPE by
1 to
    if (_fiter4>_f1iter2+999) BREAK                         ; get all two-digit FTYPE for current
FTYPE in
    if (_f1iter2<5000 | _f1iter2>5999)                                                  ; Loop
21.
    LOOP _liter8=1,9,1                                    ;^Begin Loop 23: Cycles through Lanes.

      LOOP _aiter7=100000,599999,10000                    ;^Begin Loop 24: Cycles through ATYPE in
order
        _ftotiter=_aiter7+_fiter4+_liter8                   ; to generate total X by single digit
FTYPE.
        _ftotals=_ftotals+_wffspd[_ftotiter]
        _ftdist=_ftdist+_dmiles[_ftotiter]
      ENDLOOP                                             ;^End Loop 24.

    ENDLOOP                                               ;^End Loop 23.
    endif
  ENDLOOP                                                 ;^End Loop 22.
  _supertotal=_supertotal+_ftotals                        ;^Generate overall total for all single
digit ATYPE
  _superdist=_superdist+_ftdist                           ; by all single digit FTYPE.

  if (_ftdist>0)
    _ftspd=_ftotals/_ftdist
  else
    _ftspd=0
  endif
  print list="\\"," ",_ftspd(10.2C)," ", PRINTO=1
ENDLOOP                                                   ;^End Loop 21.

if (_superdist>0)
  _superspd=_supertotal/_superdist
else
  _superspd=0
endif
_totalffspd=_superspd
print list="\\"," ",_superspd(10.2C), PRINTO=1
print list=" ", PRINTO=1
;*************************************************************************
; END FREE FLOW SPEED REPORT
;*************************************************************************

;=============================================================================================
; BEGIN CONGESTED SPEED REPORT ------------ X = Congested Speeds
;=============================================================================================
Print list=" ", PRINTO=1
Print
list="*************************************************************************************
***********************************", PRINTO=1
Print                                                                                 list="*
*", PRINTO=1
Print list="*                                                          Congested Speed (MPH)
*", PRINTO=1
Print                                                                                 list="*
*", PRINTO=1
Print
list="*************************************************************************************
***********************************", PRINTO=1
Print list=" ", PRINTO=1
;-----------------2-DIGIT FACILITY TYPES BY 2-DIGIT AREA TYPES--------------------
LOOP _a1iter=100000,599999,100000                         ;^Begin Loop 1: Cycles through Area Types
(ATYPE) by 10
  _aat1=int(_a1iter/100000)                                ; in order to get single digit ATYPE.
```

```
  print list= "Area Type ",_aat1(1.0),"x Range:",
           "\n ", PRINTO=1

  LOOP _aiter=_a1iter,599999,10000                          ;^Begin Loop 2: Cycles through ATYPE by
1
    if (_aiter>_a1iter+99999) BREAK                         ; in order to get two-digit ATYPE.
    _aat2=int(_aiter/10000)

    _avcheck=0                                              ;^Initialize ATYPE X checking variable.
    _avdist=0

    LOOP _achkiter=_aiter,599999,1                          ;^Begin Loop 3: Cycles through Lanes and
Facility Types (FTYPE)
      if (_achkiter>_aiter+9999) BREAK                      ; for current ATYPE in Loop 2 and totals
X checking variable.
      _avcheck=_avcheck+_wcgspd[_achkiter]
    ENDLOOP                                                 ;^End Loop 3.

    if (_avcheck>0)                                         ;^Begin Condition 1: If current ATYPE in
Loop 2
                                                            ; has X>0 continue to report X. Else skip
ATYPE.
      _supertotal=0                                         ;^Initialize ATYPE total X.
      _superdist=0

      Print list= "Area Type ",_aat2(2.0), PRINTO=1         ;^Header
      Print list= "                                                    Number of Lanes per Direction
", PRINTO=1
      Print list= "FType          1           2           3           4           5           6
7           8           9       Totals", PRINTO=1
      Print list= "-------------------------------------------------------------------------------------
-----------------------------------------", PRINTO=1

      LOOP _fiter=100,9900,100                              ;^Begin Loop 4: Cycles through FTYPE
                                                            ; by 1 in order to get two-digit FTYPE.
       _vcheck=0                                            ;^Initialize FTYPE X checking variable.
       _vdist=0

       LOOP _liter=1,9,1                                    ;^Begin Loop 5: Cycles through Lanes for
current
         _vcheck=_vcheck+_wcgspd[_aiter+_fiter+_liter]     ; FTYPE in Loop 4 and totals X checking
variable.
       ENDLOOP                                              ;^End Loop 5.

        if (_vcheck>0 & (_fiter<5000 | _fiter>5999))        ;^Begin Condition 2: If current FTYPE in
Loop 4
          _fft2=int(_fiter/100)                            ; has X>0 continue to report X. Else skip
FTYPE.
          print list= _fft2(2.0),"    ", PRINTO=1
          _totvols=0                                        ;^Initialize FTYPE total X.
          _totdist=0

          LOOP _liter2=1,9,1                                ;^Begin Loop 6: Cycles through Lanes to
generate ATYPE by FTYPE by Lanes total X.
            if (_dmiles[_aiter+_fiter+_liter2]>0)
              _spdspd=_wcgspd[_aiter+_fiter+_liter2]/_dmiles[_aiter+_fiter+_liter2]
            else
              _spdspd=0
            endif
            print list="\\"," ",_spdspd(10.2C)," ", PRINTO=1
            _totvols=_totvols+_wcgspd[_aiter+_fiter+_liter2]
            _totdist=_totdist+_dmiles[_aiter+_fiter+_liter2]
           _supertotal=_supertotal+_wcgspd[_aiter+_fiter+_liter2]
           _superdist=_superdist+_dmiles[_aiter+_fiter+_liter2]

          ENDLOOP                                           ;^End Loop 6
            if (_totdist>0)
              _totspd=_totvols/_totdist
            else
              _totspd=0
            endif
```

```
            print list="\\"," ",_totspd(10.2C), PRINTO=1
        endif                                                ;^End Condition 2.

      ENDLOOP                                                ;^End Loop 4.

      Print list= "---------------------------------------------------------------------------------
-----------------------------------------", PRINTO=1
      print list="Totals", PRINTO=1

      LOOP _liter3=1,9,1                                    ;^Begin Loop 7: Cycles through Lanes for
                                                             ; current ATYPE in Loop 2.
        _lntotals=0                                          ;^Initialize Lane total X.
        _lndist=0

        LOOP _aiter2=_aiter,599999,100                      ;^Begin Loop 8: Cycles through FTYPE for
current ATYPE
          if (_aiter2>_aiter+9999) BREAK                     ; in Loop 2 to generate Lane total X.
          if (_aiter2<_aiter+5000 | _aiter2>_aiter+5999)
            _lntotiter=_aiter2+_liter3
            _lntotals=_lntotals+_wcgspd[_lntotiter]
            _lndist=_lndist+_dmiles[_lntotiter]
          endif
        ENDLOOP                                              ;^End Loop 8

        if (_lndist>0)
          _lnspd=_lntotals/_lndist
        else
          _lnspd=0
        endif
        print list="\\"," ",_lnspd(10.2C)," ", PRINTO=1
      ENDLOOP                                                ;^End Loop 7

      if (_superdist>0)
        _superspd=_supertotal/_superdist
      else
        _superspd=0
      endif
      print list="\\"," ",_superspd(10.2C), PRINTO=1
      print list=" ", PRINTO=1
    endif                                                    ;^End Condition 1.

  ENDLOOP                                                    ;^End Loop 2.

  print list=" ", PRINTO=1
ENDLOOP                                                      ;^End Loop 1.

;-----------------2-DIGIT FACILITY TYPES BY TOTAL AREA TYPES--------------------

Print list= "Total Area Types ", PRINTO=1                    ;^Header
Print list= "                                                        Number of Lanes per Direction
", PRINTO=1
Print list= "FType         1          2          3          4          5          6          7
8          9       Totals", PRINTO=1
Print list= "---------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1

LOOP _fiter2=100,9900,100                                   ;^Begin Loop 9: Cycles through FTYPES to
get
  _fft2=int(_fiter2/100)                                     ; two-digit FTYPE.

  _tafvcheck=0                                               ;^Initialize FTYPE X checking variable.
  _tafvdist=0

 if (_fft2<50 | _fft2>59)
  LOOP _liter5=1,9,1                                          ;^Begin Loop 10: Cycles through Lanes
for current
                                                             ; FTYPE in Loop 9.
    LOOP _aiter4= 100000,599999,10000                         ;^Begin Loop 11: Cycles through ATYPE
for
      _tafvcheck=_tafvcheck+_wcgspd[_aiter4+_fiter2+_liter5] ; current Lanes and FTYPE in order to
total X checking variable.
```

```
      _tafvdist=_tafvdist+_dmiles[_aiter4+_fiter2+_liter5]
    ENDLOOP                                                    ;^End Loop 11.

  ENDLOOP                                                     ;^End Loop 10.

  if (_tafvcheck>0)                                           ;^Begin Condition 3: If current FTYPE in
Loop 9
    print list= _fft2(2.0),"    ", PRINTO=1                   ; has X>0 continue to report X. Else skip
FTYPE.

    LOOP _liter4= 1,9,1                                        ;^Begin Loop 12: Cycles through Lanes
for current FTYPE
                                                              ; in Loop 9.
        _totftat=0                                           ;^Initialize FTYPE total X for all ATYPE.
        _totftatdist=0

      LOOP _aiter3= 100000,599999,10000                        ;^Begin Loop 13: Cycles through ATYPE
for current Lanes in Loop 12
        _totftat=_totftat+_wcgspd[_aiter3+_fiter2+_liter4] ; in order to generate total X for FTYPE
by Lane for all ATYPE.
        _totftatdist=_totftatdist+_dmiles[_aiter3+_fiter2+_liter4]
      ENDLOOP                                                 ;^End Loop 13.

      if (_totftatdist>0)
        _totftatspd=_totftat/_totftatdist
      else
        _totftatspd=0
      endif
      print list="\\"," ",_totftatspd(10.2C)," ", PRINTO=1
    ENDLOOP                                                   ;^End Loop 12.

    if (_tafvdist>0)
      _tafvspd=_tafvcheck/_tafvdist
    else
      _tafvspd=0
    endif
    print list="\\"," ",_tafvspd(10.2C), PRINTO=1
  endif                                                       ;^End Condition 3.
 endif
ENDLOOP                                                       ;^End Loop 9.

Print list= "-------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1
print list="Totals", PRINTO=1

_supertotal=0                                                 ;^Initialize all ATYPE total X.
_superdist=0

LOOP _liter6=1,9,1                                            ;^Begin Loop 14: Cycles through Lanes.

  _lntotals=0                                                 ;^Initialize total X for Lanes.
  _lndist=0

  LOOP _aiter5=100000,599999,100                               ;^Begin Loop 15: Cycles through ATYPE
and
    if ((_aiter5<105000 | _aiter5>105999) &
        (_aiter5<115000 | _aiter5>115999) &
        (_aiter5<125000 | _aiter5>125999) &
        (_aiter5<135000 | _aiter5>135999) &
        (_aiter5<145000 | _aiter5>145999) &
        (_aiter5<155000 | _aiter5>155999) &
        (_aiter5<165000 | _aiter5>165999) &
        (_aiter5<175000 | _aiter5>175999) &
        (_aiter5<185000 | _aiter5>185999) &
        (_aiter5<195000 | _aiter5>195999) &
        (_aiter5<205000 | _aiter5>205999) &
        (_aiter5<215000 | _aiter5>215999) &
        (_aiter5<225000 | _aiter5>225999) &
        (_aiter5<235000 | _aiter5>235999) &
        (_aiter5<245000 | _aiter5>245999) &
        (_aiter5<255000 | _aiter5>255999) &
```

```
          (_aiter5<265000 | _aiter5>265999) &
          (_aiter5<275000 | _aiter5>275999) &
          (_aiter5<285000 | _aiter5>285999) &
          (_aiter5<295000 | _aiter5>295999) &
          (_aiter5<305000 | _aiter5>305999) &
          (_aiter5<315000 | _aiter5>315999) &
          (_aiter5<325000 | _aiter5>325999) &
          (_aiter5<335000 | _aiter5>335999) &
          (_aiter5<345000 | _aiter5>345999) &
          (_aiter5<355000 | _aiter5>355999) &
          (_aiter5<365000 | _aiter5>365999) &
          (_aiter5<375000 | _aiter5>375999) &
          (_aiter5<385000 | _aiter5>385999) &
          (_aiter5<395000 | _aiter5>395999) &
          (_aiter5<405000 | _aiter5>405999) &
          (_aiter5<415000 | _aiter5>415999) &
          (_aiter5<425000 | _aiter5>425999) &
          (_aiter5<435000 | _aiter5>435999) &
          (_aiter5<445000 | _aiter5>445999) &
          (_aiter5<455000 | _aiter5>455999) &
          (_aiter5<465000 | _aiter5>465999) &
          (_aiter5<475000 | _aiter5>475999) &
          (_aiter5<485000 | _aiter5>485999) &
          (_aiter5<495000 | _aiter5>495999) &
          (_aiter5<505000 | _aiter5>505999) &
          (_aiter5<515000 | _aiter5>515999) &
          (_aiter5<525000 | _aiter5>525999) &
          (_aiter5<535000 | _aiter5>535999) &
          (_aiter5<545000 | _aiter5>545999) &
          (_aiter5<555000 | _aiter5>555999) &
          (_aiter5<565000 | _aiter5>565999) &
          (_aiter5<575000 | _aiter5>575999) &
          (_aiter5<585000 | _aiter5>585999) &
          (_aiter5<595000 | _aiter5>595999))

        _lntotiter=_aiter5+_liter6                          ; FTYPE in order to generate total X for
        _lntotals=_lntotals+_wcgspd[_lntotiter]              ; Lanes.
        _lndist=_lndist+_dmiles[_lntotiter]

      endif
    ENDLOOP                                                 ;^End Loop 15.

    if (_lndist>0)
      _lnspd=_lntotals/_lndist
    else
      _lnspd=0
    endif
    print list="\\"," ",_lnspd(10.2C)," ", PRINTO=1
    _supertotal=_supertotal+_lntotals                       ;^Generate total X for all ATYPE.
    _superdist=_superdist+_lndist

ENDLOOP                                                     ;^End Loop 14.
if (_superdist>0)
  _superspd=_supertotal/_superdist
else
  _superspd=0
endif
print list="\\"," ",_superspd(10.2C), PRINTO=1
print list=" ","\n ", PRINTO=1

;---------------1-DIGIT FACILITY TYPES BY 1-DIGIT AREA TYPES SUMMARY----------------

Print list= "Total Summary Area Types by Facility Types ", PRINTO=1 ;^Header
Print list= "                                                    Single Digit Facility Types
", PRINTO=1
Print list= "AType         1x          2x          3x          4x          5x          6x          7x
8x          9x      Totals", PRINTO=1
Print list= "-------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1
```

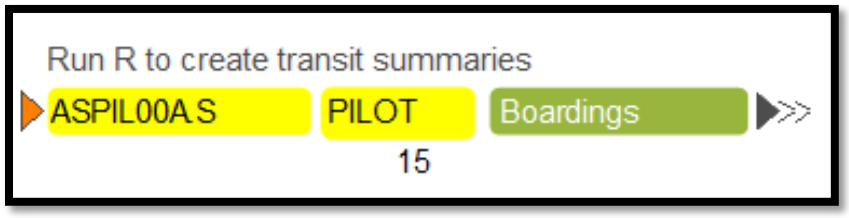
```
LOOP _a1iter2=100000,599999,100000                          ;^Begin Loop 16: Cycles through ATYPE by
10 to
  _aat1=int(_a1iter2/100000)                                 ; get single digit ATYPE.
  print list= _aat1(1.0),"x","    ", PRINTO=1

  _fttotal=0                                                 ;^Initialize total X for all ATYPE
  _ftdist=0

    LOOP _f1iter=1000,9900,1000                              ;^Begin Loop 17: Cycles through FTYPE by
10 to
                                                             ; get single digit FTYPE.
      _totftlns=0                                              ;^Initialize total X for all FTYPE by
all Lanes.
      _totftlnsdist=0
      if (_f1iter<5000 | _f1iter>5999)
      LOOP _fiter3=_f1iter,9900,100                          ;^Begin Loop 18: Cycles through two-digit
FTYPE
        if (_fiter3>_f1iter+999) BREAK                       ; for current single digit FTYPE in Loop
17.

        LOOP _aiter6=_a1iter2,599999,10000                   ;^Begin Loop 19: Cycles through two-digit
ATYPE
          if (_aiter6>_a1iter2+99999) BREAK                  ; for current single digit ATYPE in Loop
16.

            LOOP _liter7=1,9,1                                       ;^Begin Loop 20: Cycles through
Lanes for current FTYPE and ATYPE
              _totftlns=_totftlns+_wcgspd[_aiter6+_fiter3+_liter7]    ; in order to generate total
X for FTYPE by ATYPE.
              _totftlnsdist=_totftlnsdist+_dmiles[_aiter6+_fiter3+_liter7]
            ENDLOOP                                                   ;^End Loop 20.

        ENDLOOP                                              ;^End Loop 19.

      ENDLOOP                                                ;^End Loop 18.
      endif
      _fttotal=_fttotal+_totftlns                            ;^Generate total X for ATYPE.
      _ftdist=_ftdist+_totftlnsdist

      if (_totftlnsdist>0)
        _totftlnsspd=_totftlns/_totftlnsdist
      else
        _totftlnsspd=0
      endif
      print list="\\"," ",_totftlnsspd(10.2C)," ", PRINTO=1
    ENDLOOP                                                  ;^End Loop 17.

  if (_ftdist>0)
    _ftspd=_fttotal/_ftdist
  else
    _ftspd=0
  endif
  print list="\\"," ",_ftspd(10.2c), PRINTO=1
ENDLOOP                                                      ;^End Loop 16.

Print list= "-------------------------------------------------------------------------------------
----------------------------------------", PRINTO=1
print list="Totals", PRINTO=1

_supertotal=0                                                ;^Initialize overall total X.
_superdist=0

LOOP _f1iter2=1000,9900,1000                                 ;^Begin Loop 21: Cycles through FTYPE by
10
                                                             ; to get single digit FTYPE.
  _ftotals=0                                                 ;^Initialize total X by FTYPE
  _ftdist=0

  LOOP _fiter4=_f1iter2,9900,100                             ;^Begin Loop 22: Cycles through FTYPE by
1 to
```

```
    if (_fiter4>_f1iter2+999) BREAK                          ; get all two-digit FTYPE for current
FTYPE in
    if (_f1iter2<5000 | _f1iter2>5999)                                                    ; Loop
21.
    LOOP _liter8=1,9,1                                     ;^Begin Loop 23: Cycles through Lanes.

      LOOP _aiter7=100000,599999,10000                    ;^Begin Loop 24: Cycles through ATYPE in
order
        _ftotiter=_aiter7+_fiter4+_liter8                   ; to generate total X by single digit
FTYPE.
        _ftotals=_ftotals+_wcgspd[_ftotiter]
        _ftdist=_ftdist+_dmiles[_ftotiter]
      ENDLOOP                                              ;^End Loop 24.

    ENDLOOP                                                ;^End Loop 23.
    endif
  ENDLOOP                                                  ;^End Loop 22.
  _supertotal=_supertotal+_ftotals                         ;^Generate overall total for all single
digit ATYPE
  _superdist=_superdist+_ftdist                            ; by all single digit FTYPE.

  if (_ftdist>0)
    _ftspd=_ftotals/_ftdist
  else
    _ftspd=0
  endif
  print list="\\"," ",_ftspd(10.2C)," ", PRINTO=1
ENDLOOP                                                    ;^End Loop 21.

if (_superdist>0)
  _superspd=_supertotal/_superdist
else
  _superspd=0
endif
_totalcgspd=_superspd
print list="\\"," ",_superspd(10.2C), PRINTO=1
print list=" ", PRINTO=1
;*************************************************************************
; END CONGESTED SPEED REPORT
;*************************************************************************

;=========================================================================================
; BEGIN SCREENLINE SUMMARY REPORT ------------ X = SCREENLINE Volume over Count
;=========================================================================================
Print list=" ", PRINTO=1
Print
list="*****************************************************************************************
**********************************", PRINTO=1
Print                                                                                  list="*
*", PRINTO=1
Print list="*                                                    Screenline Volume over Count
*", PRINTO=1
Print                                                                                  list="*
*", PRINTO=1
Print
list="*****************************************************************************************
**********************************", PRINTO=1
Print list=" ", PRINTO=1

LOOP _sliter=1,99,1
  if (_slvol[_sliter]>0)
    Print  list=  "Screenline  ",_sliter(2.0),"  Volume/Count  Ratio:                          ",
(_slvol[_sliter]/_slcnt[_sliter])(4.2),
      "Screenline ",_sliter(2.0)," Volume:           ", _slvol[_sliter](10.0C),
      "Screenline ",_sliter(2.0)," Count:           ", _slcnt[_sliter](10.0C), PRINTO=1
   if (_sliter/5=int(_sliter/5)) Print list= " ", PRINTO=1

    if (_sliter/5=int(_sliter/5)) Print list= " ", PRINTO=1
  endif
ENDLOOP
;*************************************************************************
```

```
; END SCREENLINE SUMMARY REPORT
;*********************************************************************

;=============================================================================================
; BEGIN SUMMARY REPORT
;=============================================================================================
Print list=" ", PRINTO=1
Print
list="*************************************************************************************************
***********************************", PRINTO=1
Print                                                                                    list="*
*", PRINTO=1
Print list="*                                                                      Overall Summary
*", PRINTO=1
Print                                                                                    list="*
*", PRINTO=1
Print list=" ", PRINTO=1

print list= " Total Number of Links:          ",_numlinks(10.0C),
      "\n", " Total Lane Miles:               ",_lanemiles(10.2C),
      "\n", " Total Directional Miles:        ",_dirmiles(10.2C),
      "\n", " Total VMT using Volumes:        ",_vmtvoloncounts(10.0C), "    (Links With Counts)",
      "\n", " Total VMT using Counts:       ",_vmtcountsoncounts(10.0C), "    (Links With Counts)",
      "\n", " Total VMT Volume over Counts:  ",_vmtvolovercounts(10.2C), "    (Links With Counts)",
      "\n", " Total VHT using Volumes:        ",_vhtvoloncounts(10.0C), "    (Links With Counts)",
      "\n", " Total VHT using Counts:       ",_vhtcountsoncounts(10.C), "    (Links With Counts)",
      "\n", " Total VHT Volume over Counts:  ",_vhtvolovercounts(10.2C), "    (Links With Counts)",
      "\n", " Total Volumes All Links:        ",_totalvolumes(10.0C),
      "\n", " Total VMT All Links:            ",_totalvmt(10.0C),
      "\n", " Total VHT All Links:            ",_totalvht(10.0C),
      "\n", " Original Speed (MPH):           ",_totalffspd(10.2C),
      "\n", " Congested Speed (MPH):          ",_totalcgspd(10.2C),
  PRINTO=1
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Print list=" ", PRINTO=2
Print
list="*************************************************************************************************
***********************************", PRINTO=2
Print                                                                                    list="*
*", PRINTO=2
Print list="*                                                                      Overall Summary
*", PRINTO=2
Print                                                                                    list="*
*", PRINTO=2
Print
list="*************************************************************************************************
***********************************", PRINTO=2
Print list=" ", PRINTO=2

print list= " Total Number of Links:          ",_numlinks(10.0C),
      "\n", " Total Lane Miles:               ",_lanemiles(10.2C),
      "\n", " Total Directional Miles:        ",_dirmiles(10.2C),
      "\n", " Total VMT using Volumes:        ",_vmtvoloncounts(10.0C), "    (Links With Counts)",
      "\n", " Total VMT using Counts:       ",_vmtcountsoncounts(10.0C), "    (Links With Counts)",
      "\n", " Total VMT Volume over Counts:  ",_vmtvolovercounts(10.2C), "    (Links With Counts)",
      "\n", " Total VHT using Volumes:        ",_vhtvoloncounts(10.0C), "    (Links With Counts)",
      "\n", " Total VHT using Counts:       ",_vhtcountsoncounts(10.C), "    (Links With Counts)",
      "\n", " Total VHT Volume over Counts:  ",_vhtvolovercounts(10.2C), "    (Links With Counts)",
      "\n", " Total Volumes All Links:        ",_totalvolumes(10.0C),
      "\n", " Total VMT All Links:            ",_totalvmt(10.0C),
      "\n", " Total VHT All Links:            ",_totalvht(10.0C),
      "\n", " Original Speed (MPH):           ",_totalffspd(10.2C),
      "\n", " Congested Speed (MPH):          ",_totalcgspd(10.2C),
  PRINTO=2

; BEGIN MOE SUMMARY REPORT
;=============================================================================================
IF ({MOE}=1)

Print list=" ", PRINTO=3
```

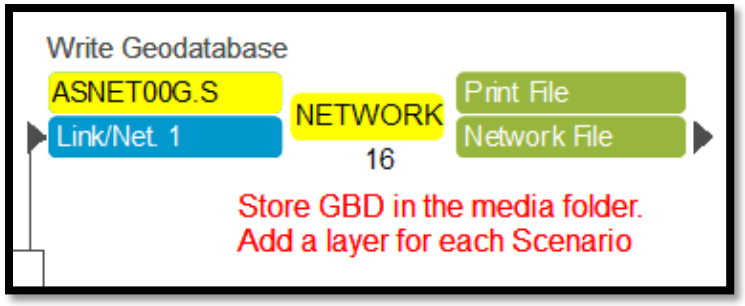
```
Print list="*****************************************************************", PRINTO=3
Print list=" DAILY MOE SUMMARY                                                  ", PRINTO=3
Print list=" {SCENARIO_FULLNAME}                                              ", PRINTO=3
Print list="*****************************************************************", PRINTO=3
Print list=" ", PRINTO=3

_VMTPC=_totalvmt/{TOTAL_POP}

LOOP _VCAPiter=1,1,1
    if (_vcap8[_VCAPiter]>0)
        Print list="MOBILTY INDEX:                    "," ",_vcap8[_VCAPiter](10.2C), PRINTO=3
        Print list=" ",PRINTO=3
    endif
   ENDLOOP
print list=
      "\n", "VEHICLE MILES OF TRAVEL:          ",
      "\n", " Total VMT:                       ","  ",_totalvmt(10.0C),
      "\n", " VMT Per Capita:                  ","  ",_VMTPC(10.2C),
      "\n", " ",
      "\n", "DEFINITION OF CORRIDORS",
;      "\n", " Corridor  0 = All other roadways",
      "\n", " Corridor  1 = Archer Rd: Tower Rd - SW13St",
      "\n", " Corridor  2 = Newberry/Univ: NW98st - NW34St",
      "\n", " Corridor  3 = Univ: NW34St - Waldo Rd",
      "\n", " Corridor  4 = SW34St: Archer Rd - Univ",
      "\n", " Corridor  5 = NW34St: Univ - NW13St",
      "\n", " Corridro  6 = SW/NWa3St: Archer Rd - NW34St",
      "\n", " Corridor  7 = Williston Rd:SW62Av - Univ",
      "\n", " Corridor  8 = Waldo Rd: Univ - NE39Av",
      "\n", " Corridor  9 = NW/NE39Av:NW98St - Waldo Rd",
      "\n", " Corridor 10 = I75:NW39Av - Williston Rd",
      "\n", " ",
      "\n", "CORRIDOR VHT:                     ",
PRINTO=3

   LOOP _crditer=1,10,1
    if (_crdvht[_crditer]>0)
        Print list="\n"," Corridor: ",_crditer(2.0),"                  ",_crdvht[_crditer](10.2C),
PRINTO=3
    endif
   ENDLOOP
  Print list=" ", PRINTO=3
  Print list="AVG DELAY/ROAD TRAVELER in minutes:",  PRINTO=3
;   LOOP _cntyiter=1,1,1
;    if (_cntyad[_cntyiter]>0)
;        Print list=" Countywide","                      ",_cntyad[_cntyiter]/{TOTAL_POP}(10.2C),
PRINTO=3
;    endif
;   ENDLOOP
;
    LOOP _urbaiter=1,1,1
    if (_urbaad[_urbaiter]>0)
        Print list=" Urban Area","                      ",_urbaad[_urbaiter]/{TOTAL_POP}(10.2C),
PRINTO=3
    endif
   ENDLOOP

    LOOP _crditer=1,10,1
    if (_crdad[_crditer]>0)
        Print                list="                 Corridor:                 ",_crditer(2.0),"
",_crdad[_crditer]/{TOTAL_POP}(10.2C),  PRINTO=3
    endif
   ENDLOOP
   Print list=" ", PRINTO=3
   Print list="TRANSIT MODE SHARE:        ",  PRINTO=3
   LOOP _cntyiter=1,1,1
    if (_cntytvol[_cntyiter]>0)

      Print   list="   Countywide:                                                ","
",(((_cntytvol[_cntyiter]/_cntytotv[_cntyiter]))*100)(10.2C),  PRINTO=3
    endif
```

```
   ENDLOOP
LOOP _urbaiter=1,1,1
    if (_urbatvol[_urbaiter]>0)

      Print   list="   Urban   Area:                                                                    ","
",(((_urbatvol[_urbaiter]/_urbatotv[_urbaiter]))*100)(10.2C),  PRINTO=3
    endif
   ENDLOOP
   LOOP _crditer=1,10,1
    if (_crdtvol[_cntyiter]>0)

      Print                  list="                  Corridor:                  ",_crditer(2.0),"
",(((_crdtvol[_crditer]/_crdtotv[_crditer]))*100)(10.2C),  PRINTO=3
    endif
   ENDLOOP

ELSE

ENDIF
;*************************************************************************
; END MOE SUMMARY REPORT
;*************************************************************************

ENDPROCESS
ENDRUN
```



ASPIL00A.S

```
FILEO PRINTO[1] = "{Scenario_Dir}\output\Transit Boarding Statistics.xlsx"
; Do not change filenames or add or remove FILEI/FILEO statements using an editor. Use
Cube/Application Manager.
;
;Inputs are not visible in the flowchart. They are:
;     {SCENARIO_DIR}\output\TLINK_OP_WALK.DBF
;     {SCENARIO_DIR}\output\TLINK_OP_AUTO.DBF
;     {SCENARIO_DIR}\output\TLINK_AM_WALK.DBF
;     {SCENARIO_DIR}\output\TLINK_AM_AUTO.DBF
;
*echo          "{CATALOG_DIR}\RSetup\R-3.5.1\bin\RScript.exe"          --vanilla          --verbose
"{CATALOG_DIR}\RSetup\Scripts\TransitStats.R"    "{SCENARIO_DIR}\output"    "{CATALOG_DIR}"    ^>
".\logFiles\TransitStats.log" >"{CATALOG_DIR}\RSetup\Scripts\runTransitSummmaryScripts.bat"
*"{CATALOG_DIR}\RSetup\Scripts\runTransitSummmaryScripts.bat"
```



ASNET00G.S

```
; Do not change filenames or add or remove FILEI/FILEO statements using an editor. Use
Cube/Application Manager.
RUN PGM=NETWORK PRNFILE="C:\GAINESVILLE\APPLICATIONS\ASNET00A.PRN" MSG='Write Geodatabase'
FILEO NETO = "{CATALOG_DIR}\media\GVL_DBD.gdb\Loads_{Scenario_ShortName}"
FILEI LINKI[1] = "{SCENARIO_DIR}\output\COMBINEDLOADED.NET"
ENDRUN
```

Appendix E: Input and Output Network Format

Input Network Format (HNET20{YEAR}.NET)
Note: {YEAR} represents the last two digits of the scenario year. The file name will be HNET2015.NET if it is base year 2015scenario or HNET2045.NET if it is 2045 Existing plus Committed scenario.

Attribute List for HNET20{YEAR}.NET

Link Attributes
A – A node
B – B node
SCRN – FSUTMS screenline code
DIR – Direction code (0=two-way, 1=one-way)
FTYPE – FSUTMS two-digit facility type. It also should be noted that any link present in the network with FTYPE=0 will not be carried through the model.
ATYPE – FSUTMS two-digit area type
LANES – Directional number of lanes
ROAD_NAME_ – Street name
ROAD_NAME2 – Alternate street name
TYPE – Represents if it is a U.S., state or county road if applicable
RCIFCLASS – RCI functional classification
DISTANCE – Link length in miles
BK_LNS – Bike lanes code (0 = no bike lanes, 1 = in street bike lanes, 2 = wide buffers for biking, 3 = off-street multipurpose facilities)
MOCF – Model output conversion factor that is found from FDOT Traffic Info DVD
AADT07 – Year 2007 two-way average annual daily traffic estimate, only for links where the count was available. For I-75 this is the sum of both directions.
COUNT07 – Directional traffic count with MOCF applied. This is used when VC (volume-to-count ratio) is calculated in the output network.
AADT10 – Year 2010 two-way average annual daily traffic estimate, only for links where the count was available. For I-75 this is the sum of both directions.
COUNT10 – Directional traffic count with MOCF applied. This is used when VC (volume-to-count ratio) is calculated in the output network.
AADT15 – Year 2015 two-way average annual daily traffic estimate, only for links where the count was available.
ADT15 – Year 2015 two-way average daily traffic estimate, only for links where the count was available.
COUNT15 – 2015 Directional traffic count with MOCF applied. This is used when VC (volume-to-count ratio) is calculated in the output network.
ONEWAY – One-way Flag (0=Two-way, 1=One-way)

POSTEDSPEED – Roadway Posted Speed (used in DLOS application)
STATION_MPO – MPO Count Station ID
EXTERNAL_CNT – External Station Node
ST_RD – State Road Flag (0= Other, 1= State Road)
SIS – Strategic Intermodal System Facilities, Florida's high priority network of transportation facilities (0= other, 1= SIS facility)
REDTRICTED – Traffic restriction flag (0= Open to traffic, 1= Bike and Ped facility)
TRUCK_RTE – Truck Route flag (1= preferred truck route)
URBANAREA – Urban area flag (0= other, 1= Urban Area, 2 = Urban Clusters)

Node Attributes
N – Node number
X – X coordinate
Y – Y coordinate
PNRDESCRIP – Bus park-and-ride lot description (text)
PNRSVCAREA – Maximum park-and-ride service area (highway access distance), in miles.
PARKINGSPA – Number of park-and-ride lot parking spaces. This value is optional because the model does not constrain the auto access mode by the number of spaces.
PNRTERMTIM – Park-and-ride terminal time (walk time from the auto to the bus stop).
KNRTERMTIM – Kiss-and-ride (auto drop-off) terminal time (walk time from the auto to the bus stop).
AMUSEFLAG – Flag to turn the lot on or off for the AM or peak network. If "1", the lot is used, if "0", the model ignores the lot.
AMPNRCOST – Cost in cents to park for AM (peak) park-and-ride trips.
MDUSEFLAG – Flag to turn the lot on or off for the MD or off-peak network. If "1", the lot is used, if "0", the model ignores the lot.
MDPNRCOST – Cost in cents to park for MD (off-peak) park-and-ride trips
SIGLOC – Traffic Signal flag (0 – No traffic signal, 1= Traffic signal-controlled intersection)

Output Network Format (COMBINEDLOADED.NET)
Note: All the input attributes that were included in the input network of HNET20{YEAR}.NET are carried over to the output network.

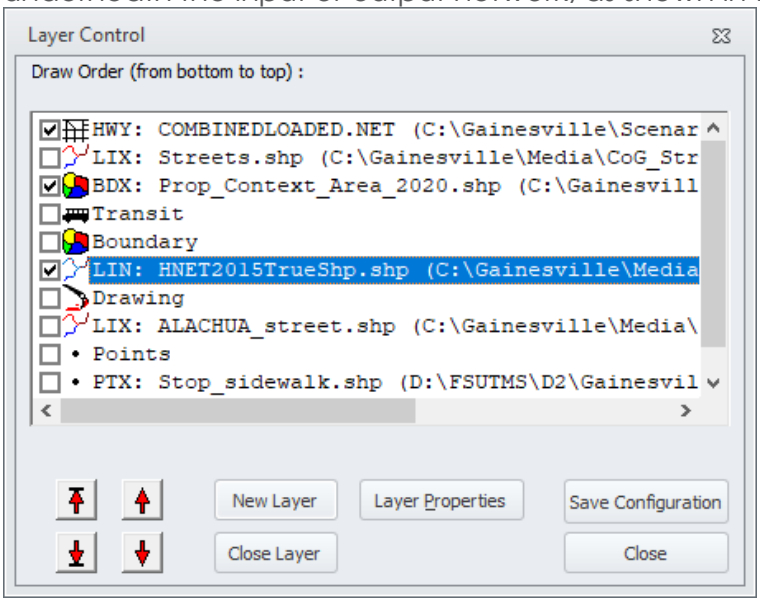
Attribute List for COMBINEDLOADED.NET
NONMOTORVOL – Total nonmotorized volumes
CGSPEED – Congested speed
CGTIME – Congested travel time (minutes)
SELZONE_MOTOR – Select zone volumes if ZoneData{YEAR}.DBF included the value of one in the SELECTZONE attribute.
UF_MOTOR – Light plus heavy vehicles with a UF trip end
LIGHTVEHICLES – Total light vehicles
HEAVYTRUCKS – Total heavy trucks
MOTORIZEDVOL – Light vehicles plus heavy trucks (MOTORIZEDVOL is most important because it is directional assigned auto volume that is used for highway evaluation)
VMT – Total motorized vehicle miles of travel.
VHT – Total motorized vehicle hours of travel.

PEDESTRIANS – Pedestrian volumes.
BICYCLISTS – Bike volumes.
VOL_CAP – Motorized volume/ (FSUTMS LOS C capacity)
DAILYCAPE– Daily FSUTMS LOS E capacity
VOL_CAPE – Motorized volume/ (FSUTMS LOS E capacity)
TranVol – Total transit volume (daily persons)
VC – 2015 Volume-to-Count Ratio (VC=MOTORIZEDVOL/COUNT15) This is only available in the base year 2015 scenario.
CONFAC – percentage of daily traffic occurring in the peak hour from VFACTORS.
CAPACITY – Hourly link capacity from the FSUTMS Speed-Capacity table, multiplied by the number of lanes
DAILYCAP – Daily capacity for roadway assignment
SPEED – Free-flow speed from the FSUTMS Speed-Capacity table. If needed, free-flow travel time in minutes can be calculated as: TIME=60*DISTANCE/SPEED
WALKTIME – Travel time in minutes for walk trips at 2.5 miles per hour.
BK_SPD – Bicycle speed
BK_TIME – Bicycle travel time in minutes

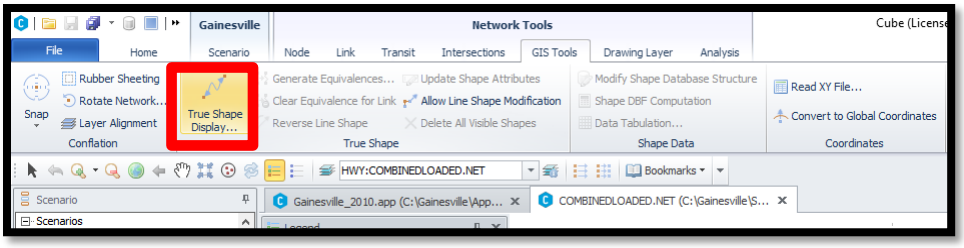
Notes Regarding True Shape Display of Networks

HNET20{YEAR}.NET and COMBINEDLOADED.NET in the Gainesville MTPO 2015 can be shown with True Shapefile Display in Cube software to be shown with curved line shapes. Please use True Shape polyline GIS shapefile which is available in model data in the following file location.
…\Gainesville\Media\Street\HNET2015TrueShp.shp

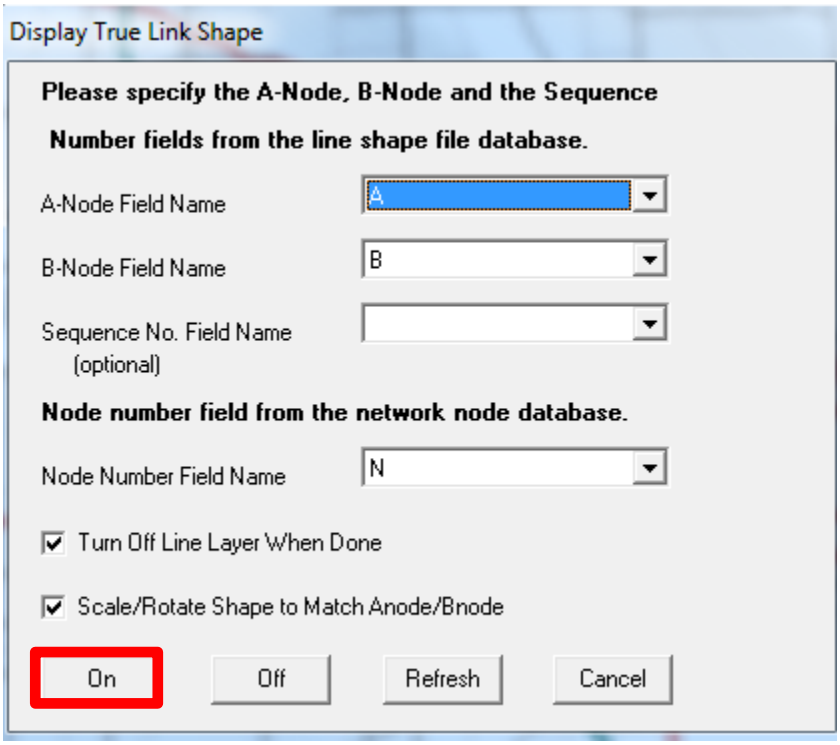
Step 1 - Make sure that True Shape GIS shapefile is correctly selected and overlaid underneath the input or output network, as shown in the next screenshot



Step 2 - Go to Tool Bar, and find and select "True Shape Display" function under "GIS Tools"



Step 3 - "A" should be selected for A-Node, "B" for B-Node and "N" for Node Number. Click **On**.



Now, the network should be displayed with curved shapes instead of straight-line shapes.

Appendix F: Glossary and Abbreviations

(**Source: Florida Standard Urban Transportation Model Structure CUBE Comprehensive Workshop Front Matter Documentation – April 2010**)

Access – Connectivity between a TAZ and the network. Access can be distinguished between highway and transit networks, and between automobile and pedestrian modes.
Advanced Traffic Management Systems (**ATMS**) **-**ATMS is the application of information and telecommunications technologies to the management of freeway and surface street facilities to maximize the use of existing roadway capacity, improve safety, reduce congestion, and provide predictable services
Advanced Traveler Information Systems (**ATIS**) **-** The collection, aggregation and dissemination of information to assist surface transportation travelers in moving from an origin to a destination.
Advanced Public Transportation Systems (**APTS**) **-** The use of information and communication technologies to improve the performance of transit services and level of service provided to customers.
Alightings - The number of persons getting off a transit vehicle.
Area Type - Network link code representing the type of land use in the area.
Attraction - The desirability of a zone. For non-home-based trips, attractions in a zone can be considered synonymous with trip destinations in that zone.
Auto Occupancy Rate - Average number of persons per vehicle.
Best Path - One of many paths between a specific origin and destination pair in a transit network determined to be the most efficient means of traveling from the origin to the destination. The default transit path methodology used in Florida.
Boardings - The number of persons getting on a transit vehicle.
Calibration - A process where models are adjusted to simulate trip-making characteristics of households in the model study area to match observed traffic activity in the study area.
Capacity - The maximum number of vehicles that can pass over a given section of a lane or roadway in one direction (or in both directions for a two-lane or three-lane highway). It is the maximum rate of flow that has a reasonable expectation of occurring. The terms "capacity" and "possible capacity "are synonymous. In the absence of a time modifier, capacity is an hourly volume. In expressing capacity, it is essential to state the prevailing roadway and traffic conditions under which the capacity is applicable. The capacity would not normally be exceeded without changing one or more of the conditions that prevail.
Centroid - Centroids are nodes used to identify the center of activity within a traffic analysis zone.
Centroid Connector - The Centroid Connector connects the traffic analysis zone centroid to the surrounding network links.
Cordon Line - An imaginary line encircling a study area. Traffic counts, travel origins and destinations, and other traffic data are collected at the locations where the imaginary line intersects the roads entering and leaving the study area. Used in modeling to estimate traffic entering and exiting the study area.

Commercial Vehicle Operations (**CVO**) **-** ITS technologies that uniquely support commercial vehicle operations to promote safe, economical, and efficient truck transportation.
Cube Voyager - A modeling software, developed by Citilabs, used as a modeling engine for the Florida Standard Model.
Demand - A desire for travel from an origin to a destination. Demand is not a fixed amount of travel, but a function of level of service.
Destination - Location to which trips are made, variously identified as a zone of specified area (in aggregate travel forecasting) or a location with a specified "attraction power," measured by things such as employees (for work trips) or square feet of sales area (for shopping trips).
Desire Line - Lines on a map representing the number of trips between zones. The thicker the line, the larger the number of trips.
EE Trips - External-External trips represent trips that travel through but have both trip ends outside of the model study area.
Facility Type - A network link code representing the type of service a roadway provides, such as principal arterial, minor arterial, collector, etc. The facility type does not always match the functional classification, as the facility type is used for modeling purposes only to simulate actual conditions.
Friction Factors (**F-Factors, FF**) **-** Reflects the regional sensitivities toward certain trip lengths for certain trip purposes. For example, home-based shopping trips may tend to be shorter than home-based work trips. Used to modify impedance during trip distribution.
Gravity Model - A mathematical model of trip distribution based on the premise that trips produced in any given area will distribute themselves in accordance with the accessibility of other areas and the opportunities they offer.
Headway - The amount of wait time between arrivals at a given transit stop for a given transit line.
Highway-Only Model - A model that only includes a roadway network thereby excluding transit.
Home-Based Trip - A trip with one end at the residence of the person making the trip.
HOV Trips - High Occupancy Vehicle trips, or carpool trips, represent the number of trips with usually two or more persons in the vehicle, including the driver.
Impedance - More general than Friction Factors, impedance shows the effect that various levels of time and cost will have on travel between zones. Impedance can include various types of time (walking, waiting, riding, etc.) and cost (fares, operating costs, tolls, parking costs, etc.). Other factors, such as comfort, convenience, personal safety, etc., may also be included.
IE Trips - Internal-External trips represent trips that have one end inside the model study area
and one end outside the model study area.
II Trips - Internal-Internal trips represent trips that have both ends inside the model study area.
Incident Management Systems - These systems manage both predicted and unexpected incidents so that the impact to the transportation network and traveler safety is minimized. Incident management involves five major phases. These are incident

detection, incident verification, incident response, incident clearance, and queue dissipation.

Intelligent Transportation Systems (**ITS**) **-** The application of information and telecommunications technologies to the management and operation of transportation systems.

IntelliDrive/Vehicle-Infrastructure Integration (**VII**) **-** The establishment of vehicle to vehicle and vehicle to roadside communication capability nationwide to enable a number of new services that provides significant safety, mobility, and commercial benefits.

Intrazonal Trip - A trip with both its origin and destination in the same zone.

Kiss-and-Ride (**KNR**) **-** A type of transit trip characterized by a transit rider being dropped off at a transit station by automobile and boarding a transit line.

Level of Service (**LOS**) **-** Multidimensional characteristics of the transportation service provided that are usually identified specifically by the location of the origin and destination of a trip and that are divided into those that are quantifiable (travel time, travel cost, number of transfers) and those that are difficult to quantify (comfort, mode image).

Link - A basic component of a network representing a segment of roadway. This component is a primary unit of analysis and carries data pertaining to roadway characteristics, traffic volumes, and performance measures.

Managed Lanes - Managed lanes help maximize the use of existing highway capacity by using price and/or occupancy restrictions to manage the number of vehicles traveling on them. Managed lanes maintain volumes consistent with acceptable levels of service even during peak travel periods.

Micro-coding - A transit modeling technique used to introduce a higher level of detail at transit stations by separating access points between modes and introducing links connecting them. Allows a more realistic representation of transferring between modes.

Mode Choice - Mode choice models calculate which trips will use the highway network and which will use the transit network. The model predicts how the trips will be divided among variable modes of travel.

Mode of Travel - Means of travel such as auto driver, vehicle passenger, mass transit passenger, walking or bicycle.

Nested Logit Model (**NLM**) **-** Analytical form for demand modeling that is suited to modeling of multiple travel choice situations by grouping different modes of travel according to their likelihood for direct competition.

Network - Set of nodes and connecting links that represent transportation facilities in an area.

Attributes normally associated with links are distances, levels of service, capacities, and volumes.

Node - A point where two links join in a network, usually representing a decision point for route choice but sometimes indicating only a change in some important link attribute.

Occupancy Model - Converts person trips to vehicle trips using auto occupancy factors.

Origin - The location of the beginning of a trip or the zone in which a trip begins.

Park-and-Ride (**PNR**) **-** A type of transit trip characterized by the act of parking at a transit station and boarding a transit line.

Path - A set of links representing a possible route between an origin and a destination. There can be a number of paths between any specific origin and destination pair.

Peak Period - The period during which the maximum amount of travel occurs. This may be one or more hours. Generally, there is a morning peak and an afternoon peak and traffic assignments may be made for each period.
Productions - The number of home-based trip ends in the zone of residence. For all non-home based trips, productions are synonymous with origins.
Ramp Metering - The application of signal control devices to regulate the number of and/or how vehicles merge into the freeway mainline lanes with the objective in most cases to balance flow and demand.
Ridership - Number of individuals using a transit line. Used as an assessment of a transit line's attractiveness.
RMSE - Root Mean Square Error is a measure of total error defined as the square root of the sum of the variance and the square of the bias. It assumes that larger forecast errors are of greater importance than smaller ones; hence they are given a more than proportionate penalty.
Road-Weather Information Systems (**RWIS**) **-** RWIS provides information to travelers and also to agencies for better deployment of resources. They use combinations of weather information services and data collected from environmental sensors.
Screenline - An imaginary line, usually along a physical barrier such as a river or railroad tracks, splitting the study area into parts. Traffic counts and possibly interviews are conducted along this line, and the crossings are compared to those calculated from the home interview data as a check of survey accuracy. Crossing may also be compared with model estimates as part of calibration.
Selected Link Analysis - Traces the entire length of each trip passing through a particular link or set of links along the network to determine where such trips are coming from and going to.
Selected Zone Analysis - Traces the entire length of each trip traveling to or from a particular zone or set of zones.
Shortest Path - A path representing the least cost option of traveling between any specific origin and destination pair.
Signal Preemption - Traffic signal preemption is a type of system that allows the normal operation of traffic lights to be preempted, often to assist emergency vehicles. The most common use of these systems is to provide emergency vehicles priority by changing traffic signals in the path of the vehicle to green and stopping conflicting traffic.
Smart work zones (**SWZ**) **-** SWZ are automated systems that provide real-time information on work zone traffic conditions. In recent years, transportation agencies across the nation have deployed portable ITS technologies to monitor traffic and manage mobility and safety during construction and maintenance of highways.
Socioeconomic Data - Demographic data, such as household, population, and employment characteristics, that are input into the model to determine the impact on trip-making patterns.
SOV Trips - Single Occupancy Vehicle trips, or drive-alone trips, represent the number of trips with only one person in the vehicle, including the driver.
Special Generators - Concentrations of activities of such size or unusual nature to warrant special consideration in trip generation analysis.
Station - A node in the transit network that offers an opportunity for automobile access.
FSUTMS

Stop Node - A node along a transit line that represents an opportunity for boardings and alightings.
Study Area Boundary - The area that is expected to take on urban characteristics in the next 20 to 30 years (by the end of the planning period).
TAZ - Traffic Analysis Zone - a small geographic area that serves as the primary unit of analysis in a travel forecasting model.
Traffic Count - The observed number of trips collected at a specific location. Used to assist with model validation.
Transit Legs - Distinct units of a transit line representing a segment from one stop to the next.
Transit paths are built by assessing the relative costs of available transit legs.
Transit Line - A collection of transit stops arranged into a route along which public transport vehicles travel. A system of interacting transit lines is a transit network
Transit Signal Priority - Transit Signal Priority (TSP) is an operational strategy that facilitates the movement of in-service transit vehicles through traffic signal controlled intersections. Signal priority modifies the normal signal operation process to better accommodate transit vehicles.
Transportation Model - A mathematical description of a transportation system's characteristics including traffic volumes, and use, roadway type and population. After a mathematical relationship is established, the model is used to predict traffic volumes based on anticipated changes in the other characteristics.
Trip Assignment - The process of determining route or routes of travel and allocating the zoneto-
zone trips to these routes.
Trip Distribution - The process by which the movement of trips between zones is estimated. The data for each distribution may be measured or estimated by a growth factor process, or by synthetic model.
Trip End - Either a trip origin or a trip destination.
Trip Generation - A general term describing the analysis and application of the relationships that exist among the trip makers, the urban area, and trip making. It is used to determine the number of trip ends in any part of the urban area.
Trip Purpose - The reason for making a trip, normally one of several possible purposes. Each trip may have a purpose at each end; (e.g., home to work) or may be classified by the purpose at the non-home end (e.g. home to shop).
Trip Table - A table showing trips between zones -- either directionally or total two-way. The
trips may be separated by mode, by purpose, by time period, by vehicle type, or other classification.
Trip Rate - The average number of trips per household for specific trip purposes. In Florida, trip rates are usually applied by household size and auto availability within each zone by trip purpose.
Validation - The procedure used to adjust models to simulate base year traffic conditions. A preliminary step that must be undertaken before models may be reasonably used to forecast future traffic conditions.
VHT - Vehicle hours of travel.
VMT - Vehicle miles of travel.

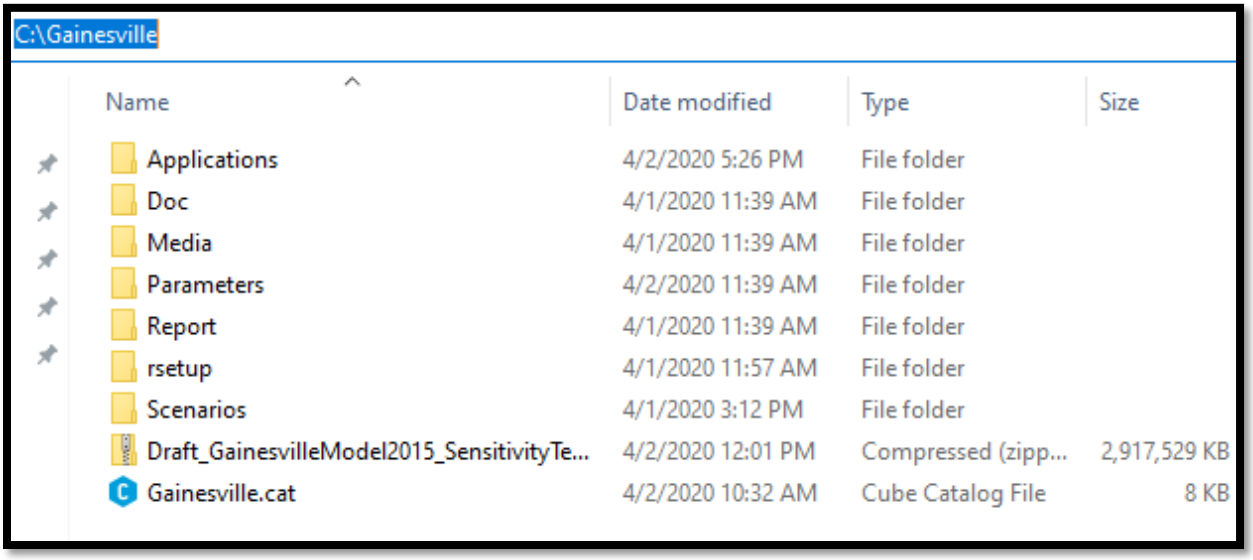
Volume-to-Capacity Ratio - The number of trips simulated in the model divided by the capacity of the link. A volume-to-capacity ratio of 1.0 represents 100 percent of the capacity.
Volume-to-Count Ratio - The number of trips simulated in the model divided by the count on the link. A volume-to-count ratio of 1.0 represents an exact match between the simulated volumes and the observed counts. Typically assessed only during validation.

ACRONYMS (Sourced from FDOT Project Forecasting Handbook 2002)
ADT - Average Daily Traffic
AADT - Annual Average Daily Traffic
D-factor - Directional traffic split
D30 factor - Proportion of traffic in the peak direction for the 30th highest hour
DHV - Design Hour Volume
DDHV - Directional Design Hour Volume
DHT - Design Hour Truck Percentage
ESAL - Equivalent Single Axle Load
FDOT - Florida Department of Transportation
FHWA - Federal Highway Administration
FM - Financial Management
FPI - Financial Project Identifier
FSUTMS - Florida Standard Urban Transportation Model Structure computer program
HCM - Highway Capacity Manual
K30 factor - Ratio of design hour volume to annual average daily trips for the 30th highest hour
Lf - Lane Factor
LGCP - Local Government Comprehensive Plan
LOS - Level of Service
MOCF - Model Output Conversion Factor
MPO - Metropolitan Planning Organization
PD&E - Project Development and Environment
PHF - Peak Hour Factor
PTMS - Portable Traffic Monitoring Site
PSWADT - Peak Season Weekday Average Daily Traffic
RCI - Roadway Characteristics Inventory database
SF - Seasonal Factor
T - Truck Factor
TCI - Traffic Characteristics Inventory database
TTMS - Telemetric Traffic Monitoring Site
V/C - Volume to Capacity Ratio
WPA - Work Program Administration
WPI - Work Program Item (First 6 digits of financial project identifier)

Appendix G: Scenario Manager / Running the Model

The model has been calibrated to 2015 conditions using the 2015 Socioeconomic data developed in coordination with the 2045 Long-Range Transportation Plan working committee members. The model volumes have been validated against the 2015 traffic counts obtained from Florida Department of Transportation and the local counts obtained from the City of Gainesville.

- Download the latest Gainesville Model using the link provided
- Unzip the folder to C:\Gainesville



Gainesville Model Folder Structure:

Applications: It hosts the cube applications and corresponding cube scripts

Doc: Placeholder for 2015 Model Development Reports

Media: Placeholder for 2015 data – traffic counts, shapefiles, transit ridership, transit routes and stops

Parameters: Model input parameters files like VFACTORS.csv, SPDCAP.dbf etc.

Reports: Cube based reporting scripts (placeholder)

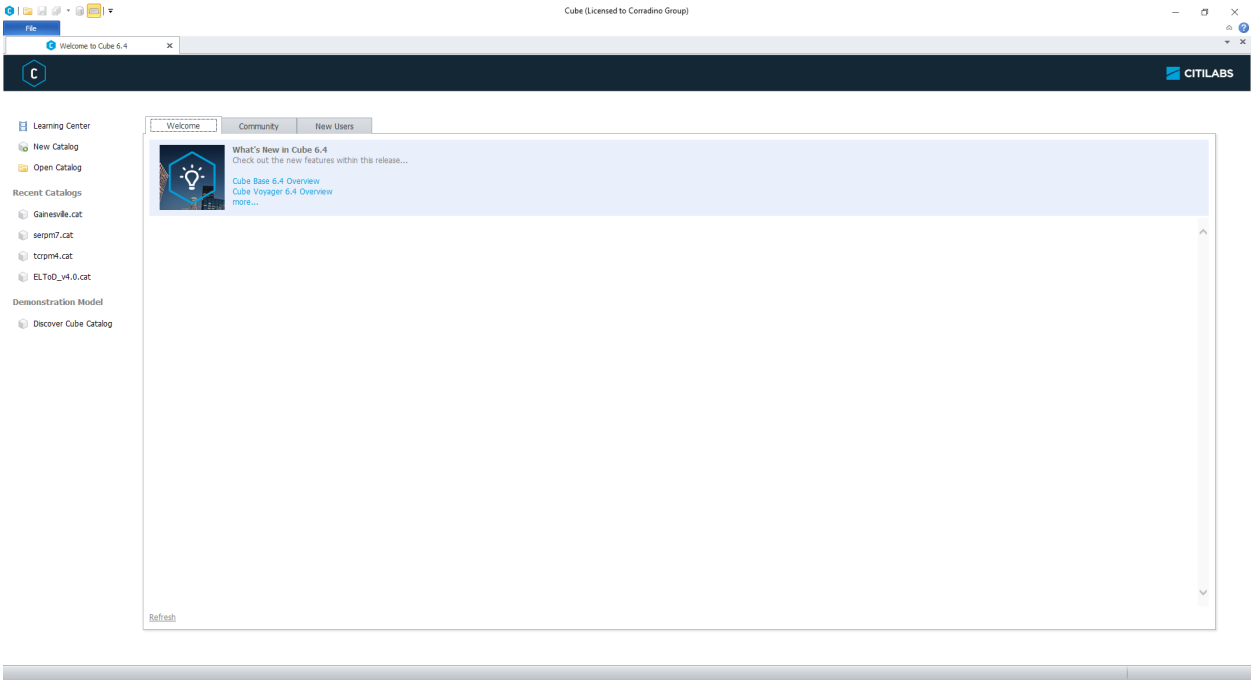
rsetup: It hosts the entire R program related dependencies and scripts for generating transit summaries and R-based Model Dashboard.

Scenarios: It hosts the scenario specific folders and corresponding input/output folders

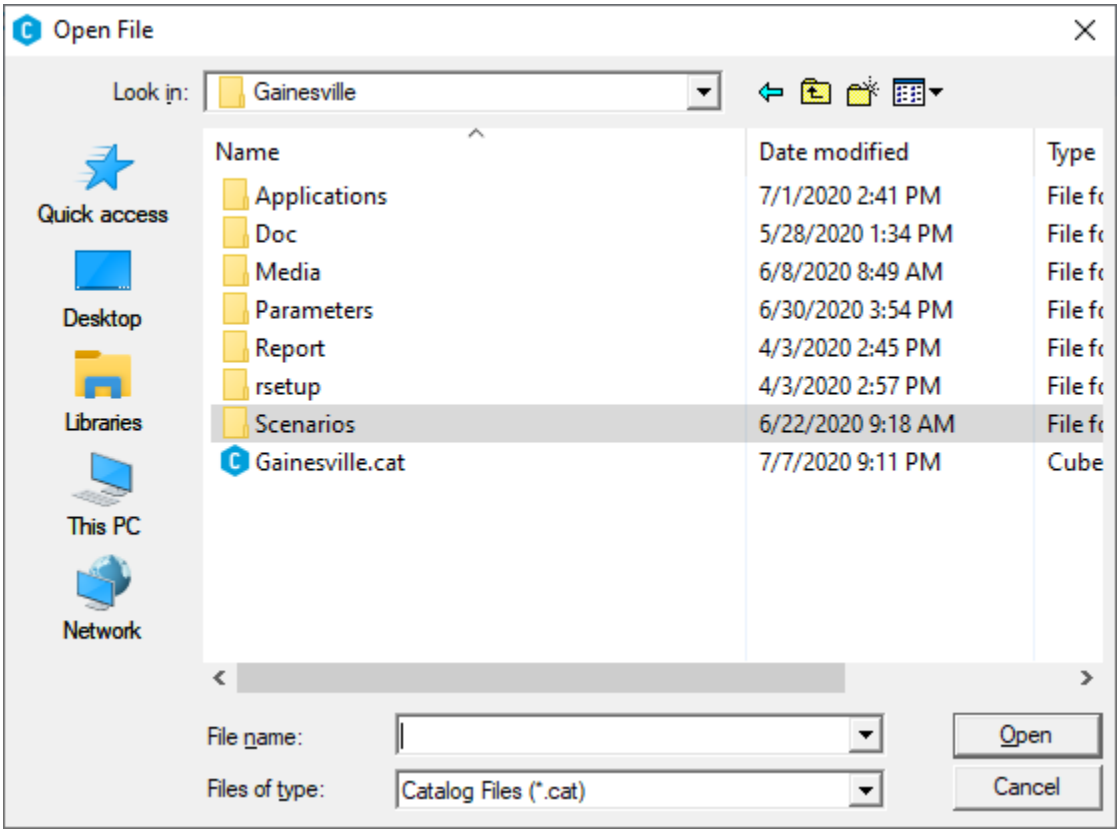
The following steps guides through the model setup procedure:

1. To run the Alachua County model, you first have to open up the Cube Application window by navigating to it or clicking the Cube desktop icon.

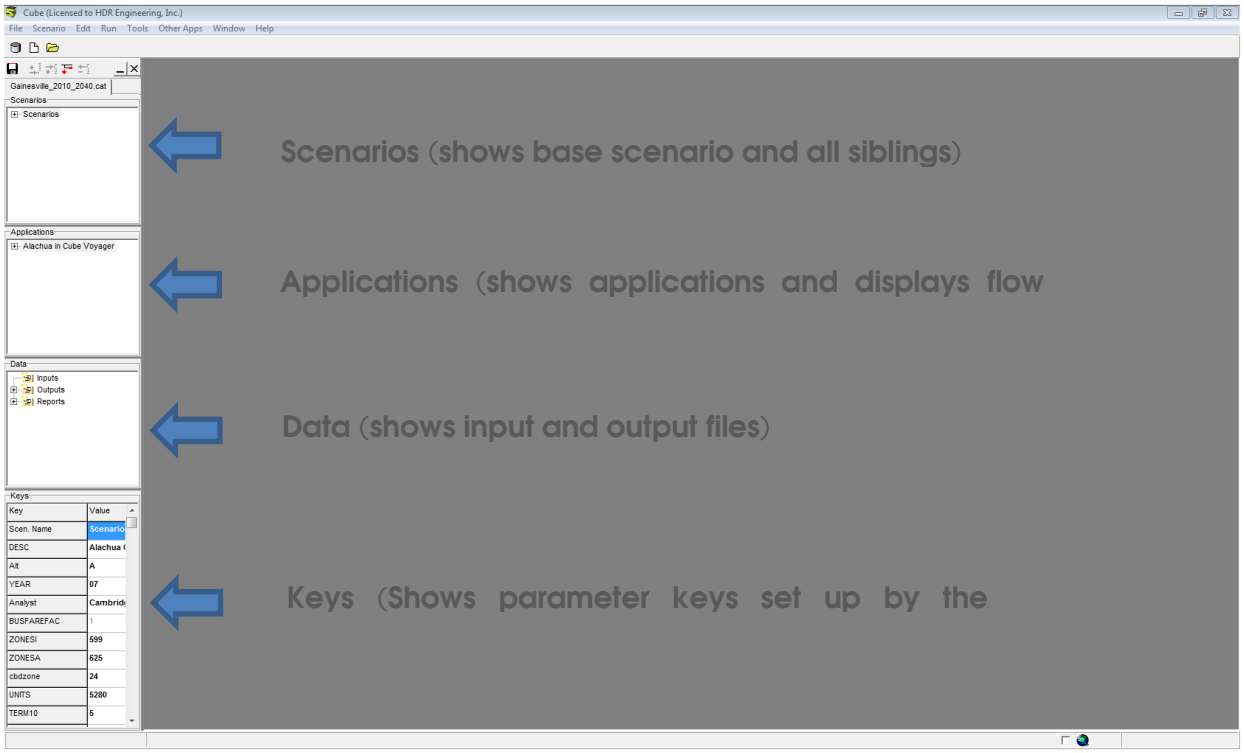
2. This brings up a dialog asking for the catalog location of the model.



3. Navigate to the location for the catalog on your computer/network and open it. The second way to open the model is to double click **Gainesville.cat** in Windows Explorer.

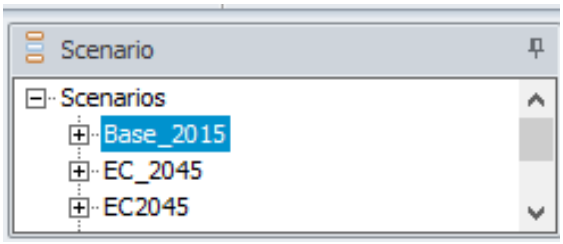


4. This brings up the Gainesville model application window.

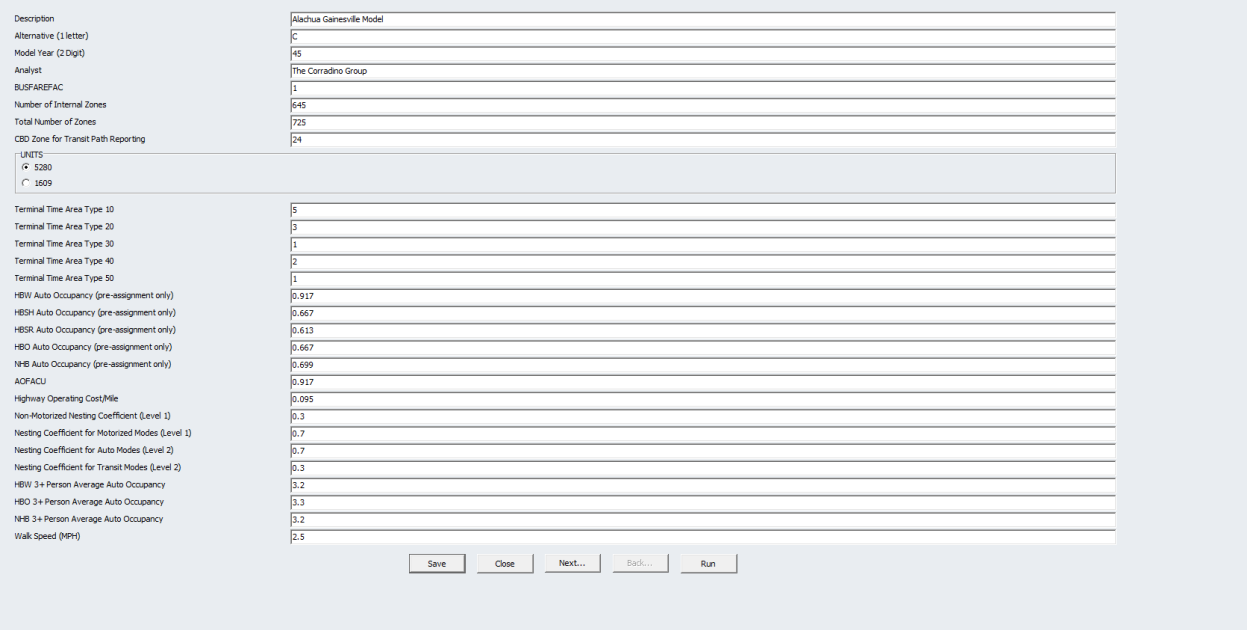


The figure above shows the various parts of the Cube catalog application window.

5. To run a particular scenario, for example the Base Year, you would select **Base 2015 in the Scenarios section of the catalog.**



When a scenario is double-clicked, the Scenario dialog box is displayed as illustrated below.



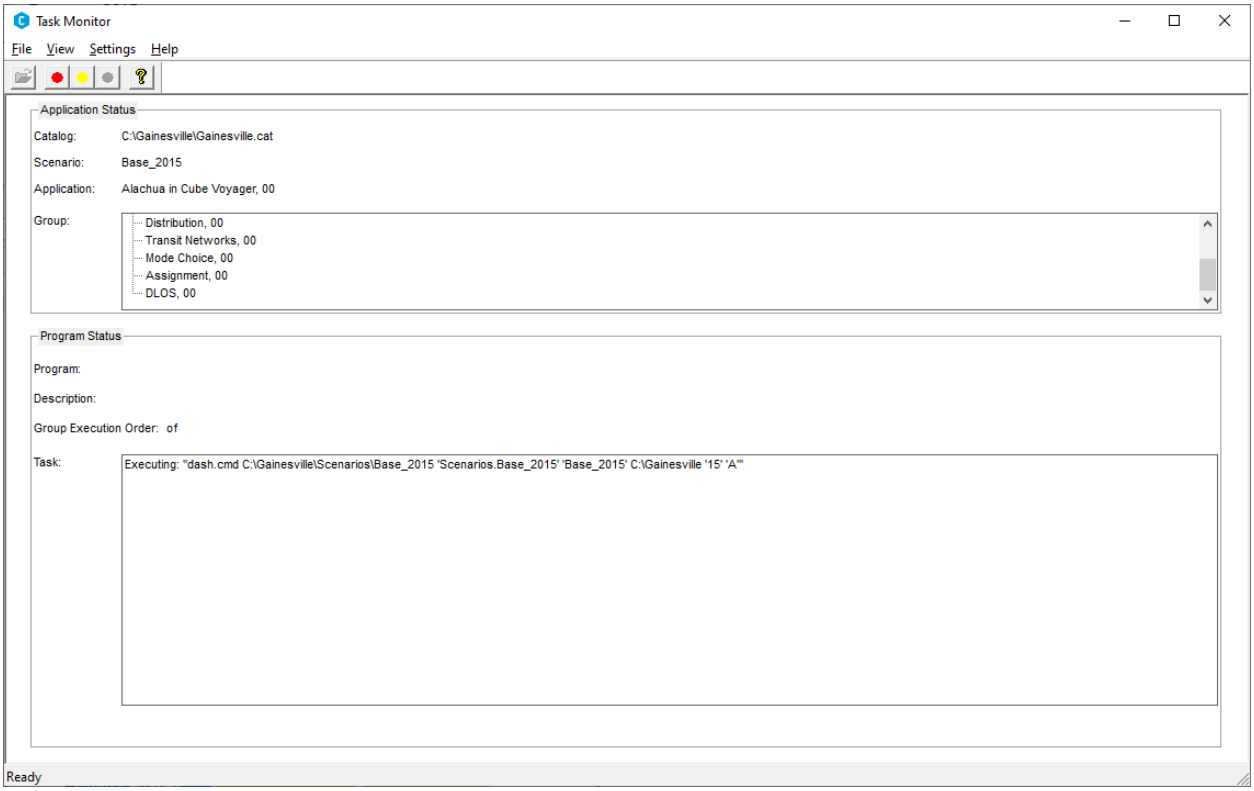
This dialog box allows you to change model runtime options and set model parameters. These parameters are stored as Cube Catalog Keys which are variables which are referenced in the model script during the relevant model processing phase. The Catalog Key values can also be observed in Part 4 of the Catalog File diagram above.

6. To run the model with the indicated variables, simply press **Run** and then **OK** and the model will be launched. When the model run is completed, press **OK.** You may then analyze the results.

The **Applications** section of the Catalog file shows the model processes and may be used to open the Cube Application Flowchart for a specific step or for the overall model.

The **Data** section of the Catalog file shows the input files, output files and any reports generated by the model run processes.

For further details on the Scenario Manager and running Florida Standard Urban Transportation Model Structure models, you may refer to the Florida Standard Urban Transportation Model Structure Comprehensive Modeling Workshops held periodically throughout the year in various locations across the state.


Task Monitor
File View Settings Help
Application Status
Catalog: C:\Gainesville\Gainesville.cat
Scenario: Base_2015
Application: Alachua in Cube Voyager, 00
Group:
Distribution, 00
Transit Networks, 00
Mode Choice, 00
Assignment, 00
DLOS, 00
Program Status
Program:
Description:
Group Execution Order: of
Task:
Executing: "dash.cmd C:\Gainesville\Scenarios\Base_2015 "Scenarios.Base_2015" "Base_2015" C:\Gainesville "15" "A""
Ready