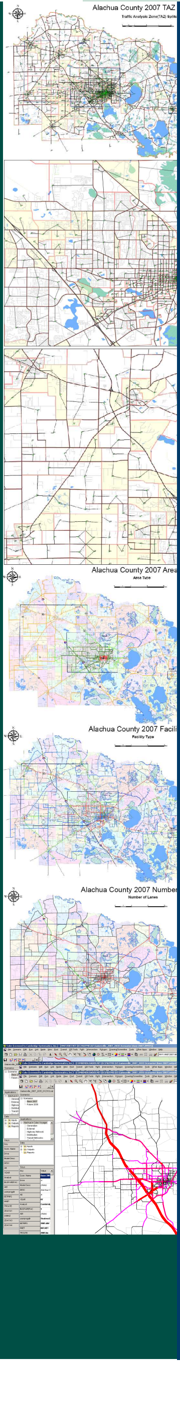
Gainesville Urbanized Area Year 2035 Long Range Transportation Plan Update

Technical Report No. 4

final

Model Update and Validation



prepared for

Metropolitan Transportation Planning Organization for the Gainesville Urbanized Area

prepared by

Cambridge Systematics, Inc.

Under subcontract to

Renaissance Planning Group

January 2011

www.camsys.com

final report

Gainesville Urbanized Area Year 2035 Long Range Transportation Plan Update

Technical Report No. 4: 2007 Model Update and Validation

prepared for

Metropolitan Transportation Planning Organization for the Gainesville Urbanized Area

prepared by

Cambridge Systematics, Inc.
1566 Village Square Boulevard, Suite 2
Tallahassee, FL 32309

January 2011

Table of Contents

1.0 Introduction **1-1**

2.0 External Trips **2-1**
- 2.1 Internal-External Trips 2-1
- 2.2 External Validation Adjustments 2-3
- 2.3 External Validation Results 2-4

3.0 Trip Generation **3-1**
- 3.1 Trip Generation Process 3-1
- 3.2 Trip Generation Validation Adjustments 3-5
- 3.3 Trip Generation Validation Results 3-5

4.0 Trip Distribution **4-1**
- 4.1 Trip Distribution Model Structure 4-1
- 4.2 Trip Distribution Model Development and Validation 4-3
- 4.3 Trip Distribution Model Results 4-4

5.0 Transit Accessibility and Path-Building **5-1**
- 5.1 Transit Access and Path-Building Model Structure 5-1
- 5.2 Transit Access and Path-Building Model Development and Validation 5-2
- 5.3 Transit Access and Path-Building Model Results 5-4

6.0 Mode Choice **6-1**
- 6.1 Mode Choice Model Structure 6-1
- 6.2 Development and Validation of Mode Choice Model 6-2
- 6.3 Mode Choice Model Results 6-3

7.0 Highway Assignment **7-1**
- 7.1 Highway Assignment Model Structure 7-1
- 7.2 Development and Validation of Highway Assignment Model 7-1
- 7.3 Highway Assignment Validation Results 7-2

8.0 Transit Assignment **8-1**

9.0 2035 E+C Scenario **9-1**

10.0 Summary and Conclusions **10-1**

Appendix A: Socioeconomic Data Format **A-1**

Appendix B: Speed Capacity Adjustments and VFACTORS **B-1**

Appendix C: Turn Penalties (TURN.PEN) **C-1**

Appendix D: Friction Factors **D-1**

Appendix E: Model Flowchart, Scripts and File Locations **E-1**

Appendix F: Input and Output Network Format **F-1**

Table of Contents Contd.

Appendix G: Glossary of Terms and Abbreviations **G-1**

Appendix H: Scenario Manager / Running Model **H-1**

List of Tables

2.1 Internal-External (IE) and External-External (EE) Percentage Splits.................... 2-3

3.1 Trip Production Rates 3-2

3.2 Attraction Rates 3-3

3.3 Dwelling Unit Weights 3-3

3.4 Special Generator Trips 3-5

3.5 Summary of Trips by Purpose 3-7

3.6 Aggregate Trip Rates 3-8

3.7 National Comparison of Person Trips per Household... 3-8

4.1 Terminal Times.. 4-2

4.2 Average Trip Lengths (in Minutes) 4-5

4.3 Intrazonal Trip Summary 4-7

5.1 Transit Access and Path-Building Model Development and Validation............. 5-2

6.1 Mode Choice Coefficients for 2007 Model.. 6-2

6.2 Mode Choice Validation Summary.. 6-4

7.1 Volume to Count Performance by Category.. 7-3

7.2 Volume to Count Performance by Screenline.. 7-6

7.3 Root Mean Squared Error.. 7-6

8.1 Year 2007 Transit Loading Estimates.. 8-2

List of Figures

2.1 External Station Locations and Model Study Area.. 2-2

4.1 Alachua County HBW Person Trip Desire Lines.. 4-6

7.1 Gainesville MTPO 2007 Model Screenlines.. 7-5

1.0 Introduction

The following Technical Report 4 (TR4) documents validation efforts on the Alachua County 2007 base year model for the Gainesville Urbanized Area Year 2035 Long Range Transportation Plan (LRTP) Update. The 2007 model is an update of the Alachua County 2000 Model for the Metropolitan Transportation Planning Organization (MTPO) for the Gainesville Urbanized Area. The 2007 model study area covers all of Alachua County, including the nine municipalities within the county. Validation efforts included updating all required input data and parameter files as well as script files for trip generation, distribution, highway and transit network development, mode choice, assignment, and reporting steps. Streamlining and clean up of script files and model structure also was completed. Finally, file formats were modified for consistency with the latest FSUTMS (Florida Standard Urban Transportation Model Structure)/Cube-Voyager[1] standards.

Technical Report 4 describes the process of reviewing highway traffic volumes and transit loadings and how input and parameter files were adjusted. When input files were modified, traffic analysis zones (TAZ), the highway and transit networks, and traffic count data were rechecked. Section 2.0 describes the update of external trips and Section 3.0 explains trip generation, while Section 4.0 discusses trip distribution. Section 5.0 describes transit accessibility and path-building process and Section 6.0 discusses the mode choice step. Section 7.0 documents highway assignment validation efforts and how the model performs and meets established FSUTMS standards. Section 8.0 describes transit assignment and resulting transit loadings. Section 9.0 explains how the future year 2035 Existing + Committed (E+C) model was created. Lastly, Section 10.0 concludes the base year validation process and related modeling efforts.

---

[1] Cambridge Systematics, Inc. *FSUTMS Cube-Voyager Data Dictionary*. Prepared for Florida Department of Transportation and Model Task Force, February 2006.

2.0 External Trips

The development of external trip input files was described earlier in Technical Report No. 3 of this series.[2] As noted in this prior document, there are 26 external zones in the 2007 Alachua County model, and these zones are now numbered 601 through 625 as depicted in Figure 2.1, which also depicts the model study area. This section of the report will focus on external trip adjustments made during model validation.

2.1 Internal-External Trips

Initial development of an external model for FSUTMS requires that external trips be divided into at least two categories: internal-over-external (IE) trips and external-to-external (EE) trips. IE trips are those trips that either have an origin outside of the study area and a destination within the study area or vice versa. EE trips have both an origin and a destination outside of the study area, but pass through the study area. There are two input files that influence external trips. These are the INTEXT and EETRIPS files with scenario year at the end of each file name. For example, the base year 2007 files are INTEXT_07A and EETRIPS_07 where the letter "A" represents an alternative name.

Although the INTEXT input file is part of the trip generation process, it is integral to generating the IE trips for the external model. The percent IE/EE splits for each external zone were derived from the I-75 Master Plan base year 2006 model. The 2006 I-75 model was validated with a focus on the I-75 corridor and appropriate adjustments as to IE/EE splits. As noted previously in Technical Report No. 3 of this documentation series, since no recent external intercept surveys were conducted, these adjustments were primarily based on logic and local knowledge. These adjustments also were conducted in an iterative manner based on impacts to volume-over-count ratios on corridors near the study boundary.

The final 2007 splits are displayed in Table 2.1. The percent IE trips were applied to the 2007 Peak Season Weekday Average Daily Traffic (PSWADT) to calculate the total number of IE trips at each external zone for the INTEXT file. The EETRIPS input file is generally the residual left after estimating IE trips in the INTEXT file. The percentage of EE trips was applied to the PSWADT by external zone and then distributed from each origin zone to each destination zone using distribution patterns from the 2006 I-75 Master Plan model.

---

[2] Cambridge Systematics, Inc. *Gainesville Urbanized Area Year 2035 Long Range Transportation Plan Update Technical Report No. 3: Data Development and Review*. Prepared for the Metropolitan Transportation Planning Organization for the Gainesville Urbanized Area, June 2010.

Figure 2.1 External station Locations and Model Study Area

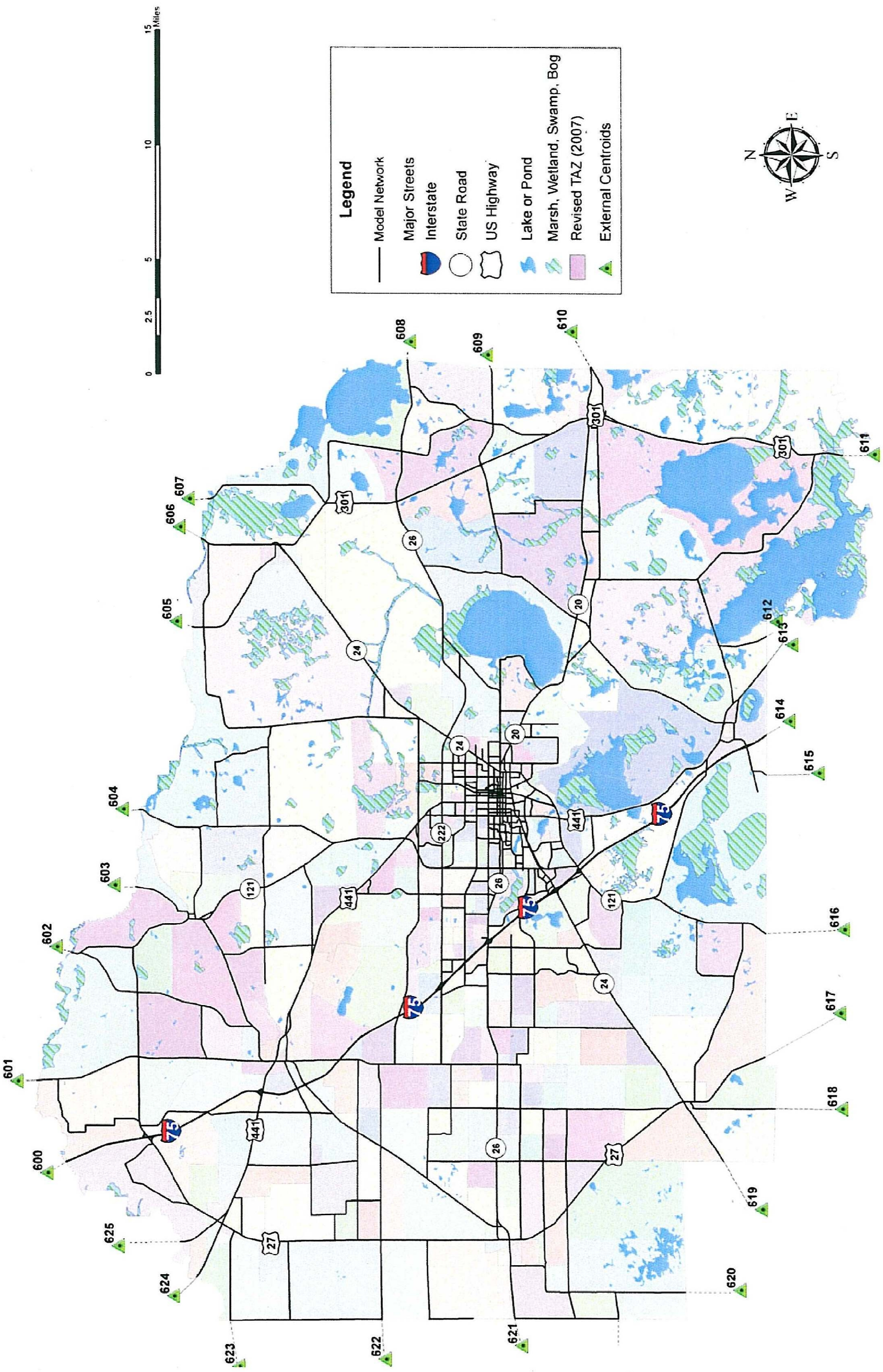


Table 2.1 Internal-External (IE) and External-External (EE) Percentage Splits

External TAZ	Total External Trips	Roadway	IE Percent	EE Percent	Total EE Trips
600	50,534	I-75 (North) at Columbia County Line	24%	76%	38,464
601	1,219	CR 241 (North) at Union County Line	68%	32%	388
602	4,747	SR 121 (North) at Union County Line	69%	31%	1,472
603	187	CR 237 (North) at Bradford County Line	86%	14%	26
604	3,618	SR 235 (North) at Bradford County Line	90%	10%	344
605	481	CR 1475 (North) at Bradford County Line	70%	30%	142
606	24,658	U.S. 301 (North) at Bradford County Line	39%	61%	14,964
607	1,124	CR 325 (North) at Bradford County Line	69%	31%	350
608	8,562	SR 26 (East) at Putnam County Line	50%	50%	4,302
609	388	CR 1474 (East) at Putnam County Line	64%	36%	138
610	9,625	SR 20 (East) at Putnam County Line	50%	50%	4,860
611	11,982	U.S. 301 (North) at Marion County Line	11%	89%	10,640
612	346	CR 225 (South) at Marion County Line	84%	16%	56
613	7,733	U.S. 441 (South) at Marion County Line	89%	11%	826
614	65,271	I-75 (South) at Marion County Line	35%	65%	42,456
615	3,657	CR 234 (South) at Marion County Line	65%	35%	1,266
616	7,785	SR 121 (South) at Levy County Line	76%	24%	1890
617	4,332	SR 45 (South) at Levy County Line	71%	29%	1,262
618	1,383	CR 241 (South) at Levy County Line	77%	23%	320
619	8,043	SR 24 (Southwest) at Levy County Line	71%	29%	2,298
620	1,323	CR 337 (South) at Levy County Line	72%	28%	370
621	9,598	SR 26 (West) at Gilchrist County Line	76%	24%	2,266
622	2,194	CR 232 (West) at Gilchrist County Line	72%	28%	610
623	4,293	NW 182 (West) at Gilchrist County Line	72%	28%	1,216
624	9,896	U.S. 27 (Northwest) at Gilchrist County Line	72%	28%	2,800
625	6,802	U.S. 441 (Northwest) at Columbia County Line	71%	29%	2,000

2.2 External Validation Adjustments

For the most part, validation adjustments to the external model consisted of modifying the INTEXT and EETRIPS files. Several iterations of the external model were executed in order to balance volumes at the external stations in such a manner as to improve model validation within the study area. Year 2007 traffic counts were used and efforts were made to achieve a 1.00 volume-over-count ratio at each external zone. Traffic counts at adjacent links were used with appropriate adjustments for the external zones of 610 and 621 since traffic counts were not available at the Alachua County line. For the external zones of 609, 612 and 620, where no nearby traffic counts were available, external volumes were prepared using previous model external volumes for the year 2000 along with year 2006 I-75 validated model external volumes.

Minor changes also were made to the INTEXT and EETRIPS files so that I-75 EE percents at external zones of 600 and 614 were increased by 10 percent to achieve a better match between model volumes and 2007 traffic counts along the I-75 mainline. Increasing the EE percent on the I-75 corridor was previously accomplished in the year 2006 I-75 Master Plan model; however, further adjustment of EE percents was needed for the year 2007 Alachua County model.

■ 2.3 External Validation Results

Model validation results are described later in Section 7.0 on the highway assignment. In particular, a review of the external cordon line indicates a reasonable match of external travel movements. The external cordon line achieves a volume-over-count ratio of 1.00. Corridors leading to or nearby external zones also were validated to satisfactory levels, based on this iterative adjustment process.

3.0 Trip Generation

The Alachua County 2007 model uses a variation on the standard FSUTMS trip generation process. Trip productions and attractions are generated by zone according to trip generation rates derived from statistical analyses of local household travel behavior. The previous 2000 model had used an external FORTRAN program called NERGEN. For the 2007 validation, Cube-Voyager scripts were instead used consistent with the latest FSUTMS procedures. This section discusses the generation process as well as efforts that were needed for the script conversion process.

3.1 Trip Generation Process

Like most FSUTMS models, the Alachua County 2007 model uses cross classification trip production rates stratified by auto availability (0, 1, 2, and 3+ auto households), dwelling unit type (single-, 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 depicted in Table 3.1. The source for these trip production rates was the North Florida Household Travel Survey, consistent with the previous 2000 Alachua County Model.

Trip attraction rates were derived from the 2005 Northeast Florida Regional Planning Model (NERPM) and are depicted in Table 3.2. It was found that borrowing these trip attraction rates led to a better balance between total productions and attractions by purpose than with using the previous Alachua County model attraction rates. Alachua County is located in the same FDOT District 2 (Northeast Florida) as the NERPM. Both model regions share a few similar socioeconomic characteristics such as less of a reliance on tourism and seasonal residents than other parts of Florida.

Dwelling unit (DU) weights were recalculated based on Census 2000 data. Census 2000 provided an accurate reflection of household distribution throughout Alachua County. The methodology used for deriving the new DU weights is the same as documented in the *FSUTMS Interactive Users' Library.*[3] In those instances where Census data demonstrated no cases within a given "Average Persons per Dwelling Unit" range, the values from the prior Gainesville 2000 model DUWEIGHT file were used. Dwelling unit weights are depicted in Table 3.3.

---

[3] Post, Buckley, Schuh, and Jernigan, Inc. *FSUTMS Interactive Users' Library CD*. Prepared for Florida Department of Transportation, 1996-1998.

Table 3.1 Trip Production Rates

Home-Based Work

Dwelling Unit Type	Number of Autos Available	Number of Persons in Household: 1	2	3	4	5+
Single-Family	0	0.35	0.64	1.01	1.50	2.08
	1	0.69	0.98	1.35	1.84	2.42
	2	1.35	1.64	2.01	2.50	3.08
	3+	1.76	2.05	2.42	2.90	3.49
Multifamily	0	0.41	0.70	1.01	1.31	1.62
	1	0.95	1.49	2.02	2.56	3.10
	2	1.65	2.30	2.95	3.60	4.25
	3+	2.21	2.89	3.59	4.27	4.96
Hotel/Motel Units		1.04	0.72	0.50	0.39	0.39

Home-Based Shopping

Dwelling Unit Type	Number of Autos Available	Number of Persons in Household: 1	2	3	4	5+
Single-Family	0	0.30	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	3.10
Multifamily	0	0.22	0.57	1.02	1.54	2.11
	1	0.50	0.95	1.40	1.83	2.27
	2	0.72	1.22	1.66	2.08	2.46
	3+	0.84	1.35	1.79	2.20	2.56
Hotel/Motel Units		0.33	1.43	2.20	2.75	3.19

Home-Based Social/Recreational

Dwelling Unit Type	Number of Autos Available	Number of Persons in Household: 1	2	3	4	5+
Single-Family	0	0.21	0.28	1.28	1.47	2.20
	1	0.48	0.85	1.43	1.31	2.37
	2	0.53	0.89	1.85	2.07	2.77
	3+	0.70	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 Units		0.66	1.81	2.97	4.29	6.49

Home-Based Other

Dwelling Unit Type	Number of Autos Available	Number of Persons in Household: 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	5.20	7.33
	3+	0.68	1.94	3.58	5.59	7.99
Multifamily	0	0.35	0.78	2.28	4.00	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 Units		0.55	1.32	2.31	3.63	4.84

Table 3.2 Attraction Rates

Purpose	Employment					Dwelling Units	School Enrollment
	Manufacturing	Other Industrial	Commercial	Service	Total		
Home-Based Work	0.00	0.00	0.00	0.00	1.80	0.50	0.00
Home-Based Shopping	0.00	0.00	0.00	0.61	0.00	0.00	0.00
Home-Based Social/Recreational	0.00	0.50	0.50	0.50	0.00	1.61	0.00
Home-Based Other	0.00	1.50	1.50	1.50	0.00	0.30	1.50
Nonhome-Based	0.00	3.54	3.54	1.71	0.00	0.30	0.00
Four-Tire Truck	0.47	0.55	0.45	0.22	0.00	0.13	0.00
Single-Unit Truck	0.12	0.15	0.13	0.04	0.00	0.05	0.00
Tractor-Trailer	0.05	0.09	0.04	0.01	0.00	0.02	0.00

Table 3.3 Dwelling Unit Weights

Average Persons Per Dwelling Unit	Percent of Households by Size[a] Category				
	One-Person Households	Two-Person Households	Three-Person Households	Four-Person Households	Five-Person + Households
0.00-1.12	0.89	0.11	0.00	0.00	0.00
1.13-1.37	0.76	0.22	0.02	0.00	0.00
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	0.00	0.05	0.07	0.14	0.74
6.00+	0.00	0.00	0.02	0.05	0.93

[a] Persons per dwelling unit.

Home-based university (HBU) and UF Campus/Dorm (DORM) trip purposes are unique to the Alachua County model. These additional purposes also were used in the Alachua County 2000 model, as it was found that this was necessary to properly model a region with a university town such as the City of Gainesville as a major trip attractor. The home-based university purpose is for trips traveling from off-campus housing to parking spaces within the UF Campus. On the other hand, the UF Campus/Dorm (DORM) trip purpose is for trips from UF 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.

Trip factors, if changed from a default of zero, are applied to productions and attractions for the HBU and DORM purposes. Trip factors are available as an adjustment tool for validation; however, the Consultant team developing the 2007 model tried to minimize the use of exogenous factors that have no basis in travel behavior theory. HBU and DORM equations were maintained the same as used in the Alachua County 2000 model, without the use of trip factors, after carefully reviewing the generated trips for these purposes. Trip production and attraction equations for the HBU and DORM purposes are listed below, as extracted from model scripts. During validation, these trip rates were relocated to the Cube catalog keys (names depicted in {brackets}) to enhance model transparency.

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 UF Parking Spaces
; Default value of {RATE_HBUA} is 1.375

UF Campus/Dorm 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

UF Campus/Dorm Attractions:
RO.HDORMUA = {RATE_HDORMUA} *ZI.1.SEATS
; SEATS is UF Classroom Seats
; Default value of {RATE_HDORMUA} is 0.7513

The ZONEDATA file format is depicted in Appendix A. ZONEDATA{YEAR}.DBF is a DBF file that combines the contents of the previous ZDATA1, ZDATA2, UF Data, and transit specific PEV (Pedestrian Environment Variable) file that were used in the Alachua County 2000 model. The ZONEDATA{YEAR}.DBF file in the Alachua County 2007 model includes population, households, employment, UF data, and PEV values.

The Alachua County 2007 model uses a total of 11 FSUTMS trip purposes:

1. Home-based work;
2. Home-based shop;
3. Home-based social/recreation;
4. Home-based other (Home-based nonwork, excluding university trips);
5. Nonhome-based;
6. Home-based university;
7. UF campus/dorm;
8. 4 tire truck;
9. Single-unit truck;
10. Tractor-trailer; and
11. Internal-external.

■ 3.2 Trip Generation Validation Adjustments

Several adjustments were made to the Alachua County 2007 FSUTMS trip generation model during validation. The use of special generators was kept to a minimum during model validation. The only special generators included were to account for a sufficient number of trips attracted to Santa Fe College and produced by dormitories within the University of Florida destined for off-campus attractors. The amount of special generator attraction trips to Santa Fe College was determined using student trip rates from the *State University System Transportation Study*[4] for the University of Florida. Special generator dormitory productions were needed for trips leaving campus as the DORM trip purpose only addresses dorm trips attracted to on-campus classrooms.

A complete listing of special generators used in the model, along with sources for trip rates, is provided in Table 3.4.

Table 3.4 Special Generator Trips[a]

TAZ	Production or Attraction	Addition or Subtraction	Person[b] Trips	Percent Trips by Purpose					Description
				HBW	HBSH	HBSR	HBO	NHB	
536	A	+	27,000	2	2	2	92	2	Santa Fe College
440	A	+	655	20	38	38	0	4	UF Dorm

4 Transportation Consulting Group. *State University System Transportation Study (BR-052) Final Report.* Prepared for the Florida State University System, August 1993.

441	A	+	576	20	38	38	0	4	UF Dorm
443	A	+	408	20	38	38	0	4	UF Dorm
449	A	+	662	20	38	38	0	4	UF Dorm
453	A	+	1,816	20	38	38	0	4	UF Dorm
460	A	+	362	20	38	38	0	4	UF Dorm

[a] Santa Fe College trip rates came from the State University System Transportation Study. A trip rate of 2.14 trips per student was used based on trip rates for UF (survey-based person trip rates were not available for Santa Fe College).

[b] Attraction trips for the Santa Fe College were derived by manually calculating trips using the noted SUS special generator trip rate and substituting manual trip generation results for machine-generated model results. Production trips for the UF dormitories were taken from the Alachua County 2000 model, assuming no growth in the number of dormitory units.

■ 3.3 Trip Generation Validation Results

Throughout the validation process, trip generation statistics were summarized to assess model validity. Comparisons were made between the Alachua County 2000 model, the Alachua County 1990 model, the Polk County 2000 model, the CRTPA (Capital Region Transportation Planning Agency, of Tallahassee, FL) 2003 model and other comparable nontourist/retiree-oriented FSUTMS models. Statistical comparisons also were made against reasonableness ranges and other models in the United States using statistics available in the *FSUTMS-Cube Framework Phase II: Model Calibration and Validation Standards Final Report*.[5] Statistics from other models were sometimes aggregated to account for different trip purpose schemes. The script files and model flowchart for the Trip Generation routine can be found in the appendices.

Table 3.5 provides a summary of trips by purpose. When compared against the Alachua County 2000 model, the Alachua County 2007 model indicates that the percent production trips by purpose remains fairly constant with a maximum difference of up to one percent. Total trip productions increased by approximately 18 percent between 2000 and 2007 models, representing growth in households, student enrollment, and employment.

Table 3.6 provides comparisons of aggregate trips per household, person, and employee, along with persons per household between the Alachua County 2007 model and several other models. Table 3.7 provides comparisons of trip rates per household with typical benchmark values and other models throughout the United States. These comparisons show that the Alachua County 2007 model is reasonably consistent with other models in terms of aggregate trip rates. Compared to other Florida models, the Alachua County 2007 model is somewhat at the high end of typical ranges; however, this seems to be reasonable given the fact that Gainesville is populated by many university students who typically generate more trips per household than nonstudent households.

[5] Cambridge Systematics, Inc. *FSUTMS-Cube Framework Phase II: Model Calibration and Validation Standards Final Report.* Prepared for Florida DOT Central Office, October 2008.

Table 3.5 Summary of Trips by Purpose

	Alachua County 2007			Alachua County 2000		1990	2003	2000
Purpose	**Productions**	**Percent by Productions**	**Unbalanced Attractions**	**Productions**	**Percent by Productions**	**Alachua County**	**CRTPA**	**Polk County**
Home-Based Work	183,299	13.96%	334,938	150,235	13.45%	19.64%	14.00%	15.73%
Home-Based Shop	132,480	10.09%	192,495	114,552	10.25%	12.70%	11.00%	9.84%
Home-Based Socrec.	119,659	9.11%	246,954	103,996	9.31%	12.81%	7.00%	9.32%
Home-Based Other	253,457	19.30%	317,271	220,197	19.71%	30.49%	30.00%	24.58%
Nonhome-Based	329,228	25.07%	329,947	286,573	25.65%	24.35%	24.00%	24.48%
Home-Based University	75,939	5.78%	22,067	52,809	4.73%			
Dormitory-Based University	23,570	1.79%	27,495	26,492	2.37%			
Truck-Taxi	81,502	6.21%		69,127	6.19%	0.00%	8.00%	8.21%
Internal-External	114,063	8.69%		93,299	8.35%	0.00%	6.00%	7.85%
Total	**1,313,197**	**100.00%**		**1,117,280**	**100.00%**	**100.00%**	**100.00%**	**100.00%**

Table 3.6 Aggregate Trip Rates

Unit of Measure	2007 Alachua County	2000 Alachua County	2003 CRTPA	2000 Polk	2000 Census *
Persons per Household	2.21	2.84	2.39	2.60	2.25
Internal Trips per Household	11.05	11.35	9.63	9.70	N/A
Internal Trips per Person	5.00	4.65	4.03	3.73	N/A
Internal Trips per Employee	9.05	7.73	7.68	8.68	N/A

* *Source: 2000 Census Data for City of Gainesville.*

Table 3.7 National Comparison of Person Trips per Household

Region	Year	Person Trip/Household
Bay County, Florida	2003	8.48
Orlando, Florida	2000	8.73
Jacksonville, Florida	2000	8.84
Polk County, Florida	2000	8.84
Tallahassee, Florida	2003	9.63
Tampa, Florida	1999	7.76
Treasure Coast Region, Florida	2000	11.28
Chattanooga, Tennessee	2000	8.05
South Bend, Indianapolis	2002	7.90
Nashville, Tennessee	2002	8.59
Memphis, Tennessee	2004	8.20
Atlanta, Georgia	2000	8.15
Charleston, South Carolina	2003	7.62
Knoxville, Tennessee	2000	8.40
Gainesville, Florida	1990	10.18
Gainesville, Florida	2000	11.35
Gainesville, Florida	2007	11.05

Sources: *FSUTMS-Cube Framework Phase II: Model Calibration and Validation Standards Final Report.* FDOT Central Office, October 2008.

4.0 Trip Distribution

Trip Distribution models link trip productions and attractions between pairs of TAZs. These interchanges are typically calculated through the application of a Gravity Model. A Gravity Model distributes trips among zones directly proportional to the relative attractiveness of each individual zone and inversely proportional to the friction between each zone (i.e., distance). The result is a matrix of person trips that is later balanced in order to be defined in terms of origins and destinations (as opposed to productions and attractions). Resulting person trip matrices are processed later in the model chain during mode choice to allocate trips by auto occupancy and transit categories and convert these to vehicle trips.

Validation of the Alachua County 2007 trip distribution model primarily involved modification of the highway and transit networks. Evaluation of the trip distribution model was accomplished by comparing statistics for average trip length and the percentage of intrazonal trips between the Alachua County 2007 model and other comparable models across Florida, including the Alachua County 2000 model and the Polk County 2000 model. Additionally, desire line maps were prepared to show travel movements between zones aggregated to districts and reviewed for general logic.

■ 4.1 Trip Distribution Model Structure

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

Building Highway Networks

FSUTMS includes a module known as “Highway Network” to construct 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 were updated to more accurately reflect 2007 conditions of the roadway system throughout Alachua County. In addition to updating existing roadway characteristics, the Consultant also added Outlying Business District area types (OBD AT 41-43) where appropriate, as this category was not included in the 2000 model network. As described in Technical Report 3, several roadways were added to the model to better reflect local travel patterns and arterial flyovers and other access controls were coded where these exist in the network. Also, the previous model speed and capacity lookup table (SPDCAP file) was replaced with one previously updated by the Consultant for use in the 2000 Polk County and NERPM models, reflecting capacities found in the latest FDOT *QualityLevel of Service Handbook.*[6]

Travel Time Skims

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

The first of these is the TCARDS file. The TCARDS file contains a record of all prohibited movements in the network. Turning movements were reviewed to include any updated prohibited movements for year 2007 conditions during validation. The TCARDS file also can 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. An input file called TOLLLINK 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 Alachua County 2007 model.

Intrazonal times represent the travel time it takes to travel within or across a zone. These times are calculated as one-half the travel time from one zone to the nearest adjacent zone. 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 a final 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 greatest in central business districts and lowest in residential areas. Table 4.1 lists the terminal times by area type used in the Alachua County 2007 model.

Table 4.1 Terminal Times

Terminal Times[a]	Area Types	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	Nonurbanized 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

[6] *Quality/Level of Service Handbook*. Florida Department of Transportation. 2002.

[a] Terminal Times listed in whole minutes.

Trip Distribution Module

The "DISTRIBUTION" module distributes trips between zones using a Gravity Model and produces a set of congested highway skims. The primary input data used for DISTRIBUTION is the friction factor (FF) file. This file is used by the Gravity Model to measure the effects of spatial separation between zones for the purposes of 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 Alachua County 2000 model were used for the Alachua County 2007 model without modification. Since no new household travel surveys were conducted since the prior 2000 model calibration and validation, and no significant errors were found in the resulting trip distribution, it was decided to maintain the sanctity of the existing friction factor set in the 2007 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 specifically from one zone in the model to another. Trips are distributed according to the 10 trip purposes found in the Alachua County 2007 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 checks, modifications, and adjustments made to trip distribution assumptions in order 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 subsequent steps in the model chain (i.e., mode choice and trip assignment). Consequently, efforts were taken to maximize the accuracy of the Alachua County 2007 trip distribution module. This effort included adjustments to network speeds and capacities and corrections of network link attributes.

Speeds and Capacities

As noted earlier, a speed and capacity lookup table, developed to be consistent with the *Quality/Level of Service Handbook* from FDOT, was used in the Alachua County 2007 model validation. After implementation of the SPDCAP file, an iterative process of manual adjustments to speeds was conducted in order to improve model validation while maintaining a logical hierarchy of speeds. Primarily, adjustments were made to be specific to certain area type/facility type combinations so as to avoid unintended impacts. Generally, speeds on the interstates were increased, and facilities other than one-way roads within CBDs and CBD fringe areas were somewhat increased. The SPDCAP adjustments are shown in Appendix B, along with the VFACTORS file that adjusts absolute capacities to practical capacities for trip assignment diversion.

Penalties and Prohibitors

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 all together. Prohibitors are confined to ramps located along the Interstate 75, mainly to guide trips to the correct ramps for each travel movement. No time penalties were added during the 2007 model validation effort. The TURN.pen file, is depicted in Appendix C.

Friction Factors

The friction factor file used in the Alachua County 2007 model is identical to the file used in the Alachua County 2000 model. Further consideration of the friction factors did not indicate a need for modification in order to improve trip distribution. Average trip lengths seemed reasonable, intrazonal percentages made sense, and aggregate trip distribution patterns looked logical. Furthermore, there were no updated household travel diary survey data for Alachua County to allow for calibration of new friction factors. A copy of the validated friction factor file (FF.dbf) is depicted in Appendix D.

Script File

The script file and flowchart screenshots for Trip Distribution, and other model steps, can be found in Appendix E. The scripts, along with the model flowchart, illustrate each step in trip distribution including input and output files and model flow parameters such as catalog keys.

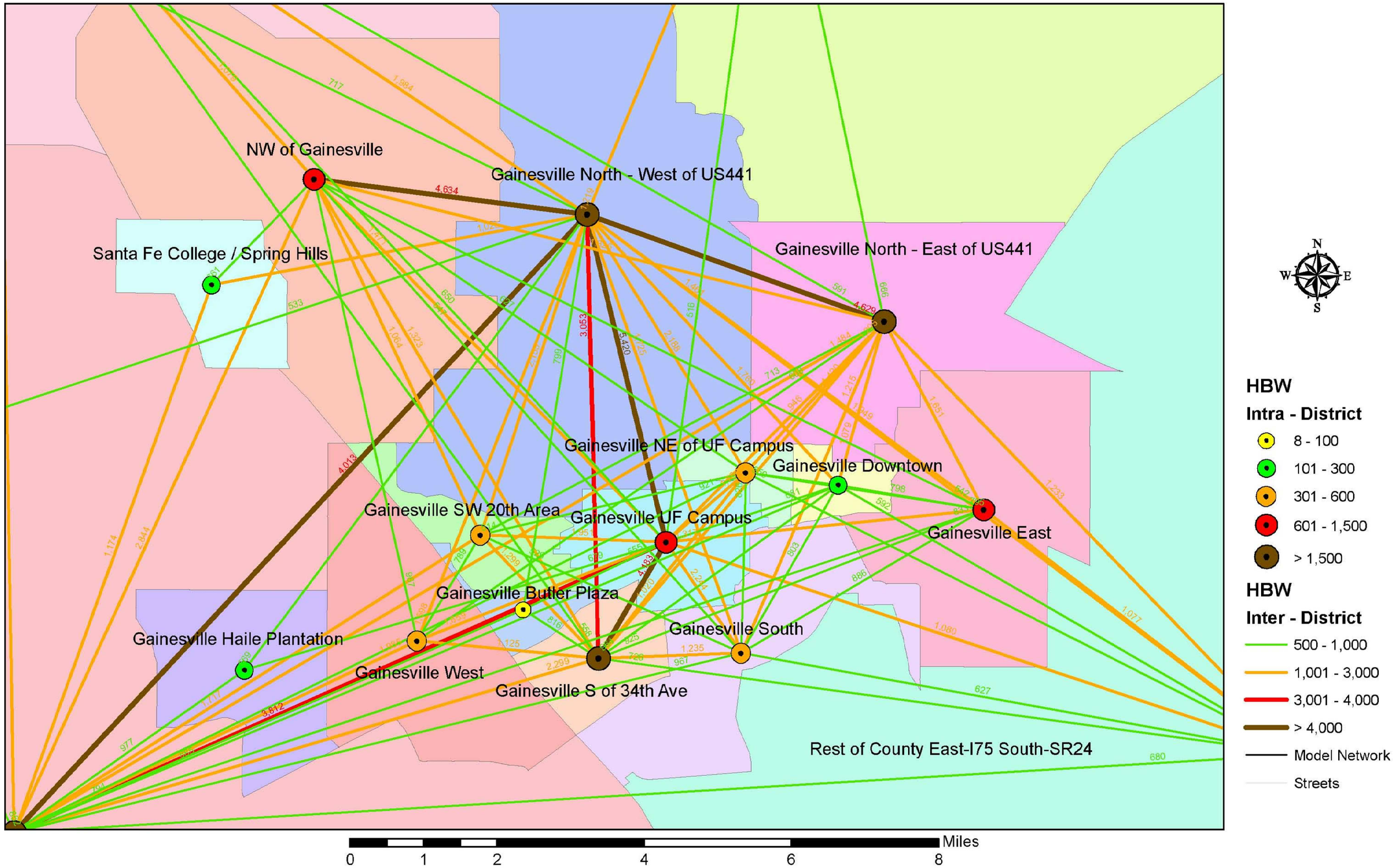
■ 4.3 Trip Distribution Model Results

The three fundamental Gravity Model checks discussed in this section are aggregate trip distribution patterns by district, the average trip length by purpose and the percentage of intrazonal trips. An analysis of volume-over-count summaries along screenlines also can be helpful in establishing the accuracy of trip distribution. However, as screenline summaries apply more significantly to the analysis of traffic assignment, these will be discussed later in Section 7.0.

Aggregate Trip Distribution Patterns by District

The Consultant requested input from the Metropolitan Transportation Planning Organization for the Gainesville Urbanized Area Long Range Transportation Plan Technical Advisory Committee Subcommittee on logical district boundaries in order to aggregate trips from zones to districts for desire line analysis. The subsequent production of desire line maps to visualize trip distribution patterns by purpose within the Alachua County 2007 model, allowed for a greater level of comfort in the reasonableness of travel movements between key subareas within the model. Figure 4.1 depicts an example desire line map for the home-based work trip purpose within the Gainesville urbanized area.

Figure 4.1 Alachua County HBW Person Trip Desire Lines



Average Trip Length by Purpose

Table 4.2 shows a comparison of average trip length statistics generated by the 2000 and 2007 Alachua County models, and several other models of comparable size throughout Florida. Comparisons between the Alachua County 2000 model and the Alachua County 2007 model show a general increase in average minutes traveled from 2000 to 2007.

Table 4.2 Average Trip Lengths (in Minutes)

Purpose	2007 Alachua County	2000 Alachua County	1990 Alachua County	2000 North Florida HH Travel Survey[a]	2007 ACS[b]	2003 CRTPA	2000 Polk County
Home-Based Work	14.73	13.92	14.74	24.60	16.20	19.80	17.03
Home-Based Shop	13.10	13.60	12.21	17.60	N/A	17.76	14.04
Home-Based Social/Recreation	12.55	11.97	11.26	18.60	N/A	17.56	15.01
Home-Based Other	13.37	12.79	11.88	20.80	N/A	18.86	15.06
Nonhome-Based	10.79	9.05	8.92	19.00	N/A	17.10	13.73
Home-Based University	9.14	8.08	N/A	N/A	N/A	N/A	N/A
UF Campus/Dorm	6.22	4.19	N/A	N/A	N/A	N/A	N/A
Truck-Taxi	15.18	13.74	11.32	N/A	N/A	16.73	15.52
Internal-External	25.69	25.78	30.01	N/A	N/A	46.20	26.17
TOTAL	**13.62**	**11.65**	**11.65**	**20.12**	N/A	**22.00**	**16.85**

[a] Source: North Florida 2000 HH Travel Survey Final Report Table 5.15 (Reported Mean).

[b] Source: 2000 American Community Survey Data for City of Gainesville.

Intrazonal Trip Distribution

Comparisons between the Alachua County 2000 model and the Alachua County 2007 model indicate that the percentages of intrazonal trips decrease for most trip purposes. This decrease is a result of zone splits made to the 2007 model. The exception to this is home-based shop, which increased by approximately 1.6 percent. These results are illustrated on Table 4.3.

Table 4.3 Intrazonal Trip Summary

	Alachua County 2007			Percent Intrazonal			
Purpose	**Total Trips**	**Intrazonal Trips**	**Percent Intrazonal**	**2000 Alachua County**	**1990 Alachua County**	**2003 CRTPA**	**2000 Polk County**
Home-Based Work	183,288	2,783	1.52%	1.91%	1.81%	0.24%	3.16%
Home-Based Shop	132,443	5,585	4.22%	2.67%	3.35%	0.45%	3.63%
Home-Based Social/Recreation	119,642	9,302	7.77%	9.51%	6.51%	0.73%	10.77%
Home-Based Other	252,428	10,971	4.35%	4.96%	5.38%	0.62%	4.16%
Nonhome-Based	329,119	23,675	7.19%	8.96%	4.73%	0.83%	4.89%
Home-Based University	75,940	39	0.05%	0.47%	N/A	N/A	N/A
UF Campus/Dorm	23,570	1,313	5.57%	2.01%	N/A	N/A	N/A
Truck-Taxi	81,199	1,212	1.49%	1.76%	5.55%	1.25%	5.71%
Internal-External	114,055	0	0.00%	0.00%	0.00%	0.00%	0.00%
Total	**1,311,684**	**54,880**	**4.18%**	**5.00%**	**4.51%**	**0.66%**	**4.93%**

5.0 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.

Transit path-building involves the generation of zone-to-zone transit paths, transit skims, transit fares, and station matrices. These files are built for each of the transit modes during peak and off-peak periods occurring in the model. The Alachua County 2007 model currently only has one mode, local bus, but has the capability of expansion to other transit modes in the future.

This section of the report provides information concerning validation of the transit access and path-building steps in the model. Script files directing the execution of building the transit network and transit paths can be found in Appendix E.

5.1 Transit Access and Path-Building Model Structure

The Alachua County 2007 model uses a nested logit mode choice transit model. The FSUTMS Transit Network module was used to establish transit access and path-building. The module constructs separate peak-period (AM) and midday (MD) transit networks using restrained highway skims as an input to represent congested zone-to-zone travel times. Transit path-building involves the generation of transit path matrices, fares, skims, and station-to-station interchanges.

Transit accessibility was represented by each zone's pedestrian environmental variables (PEV) that are stored in the ZONEDATA file, as described earlier. The PEV defines several factors that are essential to have sufficient accessibility to bus stops, such as sidewalk availability, ease of street crossing, nonmotorized connections, and building setbacks. Each variable is given a score between 0 and 3, and accumulated scores of all the four PEVs 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 PEV scores as well as updating "SUM" values to get total PEV scores for each TAZ. Table 5.1 indicates what each PEV value represents. These variables and categories remain unchanged from the 2000 model.

Table 5.1 Pedestrian Environment Variables (PEV)

	Pedestrian Environment Variable (PEV) Values			
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 with 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. In addition, walk access links were checked in order 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 year 2007 operating conditions. This required significant rerouting of the transit lines. The second was to remove some UF on-campus circulators that resulted in overassignment. Finally, PEVs in the ZONEDATA file were reviewed and updated for each zone. Walk access links that were built using the PEVs were checked for connectivity.

Transit Path-Building

Validation of transit path-building generally includes iterative adjustments to the following parameters:

- Maximum transfers;

- Maximum travel times;
- Wait times;
- Transfer times;
- Minimum and maximum wait penalties;
- Run time factors;
- Transfer penalties; and
- Maximum fare.

These parameters and other aspects about transit path-building currently reflect base year 2007 conditions. During the validation process, model estimates for transit transfers seem to be higher than 16 percent that the Regional Transit Systems (RTS) reported in their recent on-board travel survey,[7] thus the transfer penalty was adjusted.

Transit Fare

The RTS transit fare was $1.00 during the year 2007, used for the Alachua County base year 2007 model validation. This fare has since increased to $1.50, according to the RTS. Details on how the bus fare was adjusted during validation are further explained in Section 9.0. A scenario key named as BUSFAREFAC has been introduced as a bus fare factor for this model. BUSFAREFAC represents a dollar amount of transit fare, which the subsequent mode choice script uses to apply any fare change for future year scenarios. BUSFAREFAC is set to 1.0 for base year 2007, resulting in no impact to base year validation of transit trips.

Review of Transit Access and Transit Routes

As validation efforts moved towards reasonable transit assignments, bus stop locations and local zonal access to bus stops were reviewed, using the RTS bus stop GIS location file. Headway data also was provided from the RTS and the transit route file (troute07.lin) was updated. The transit route file had to be overlaid with 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 updated at the same time as the transit route file was updated. “Later Gator” bus routes that were previously included in the 2000 model were subsequently removed from the year 2007 Alachua County model during validation as these are evening bus services specifically for University of Florida students and operate for only limited hours while the model is designed to estimate daily peak and off-peak transit ridership.

---

[7] Tindale-Oliver & Associates, Inc. *Gainesville Regional Transit System 2010 to 2019 Transit Development Plan* Major Update Technical Cemorandum #1 Regional Transit System On-Board survey. Prepared for Gainesville Regional Transit System. June 2009.

As described in Technical Report No. 3 of this series, several transit "stations" were coded into the transit network, each of which allow park-and-ride access in addition to walk access. The locations where park-and-ride access was coded can be summarized as follows:

- Oaks Mall Park-and-Ride Lot – 300 spaces;
- Harn Museum – 100 spaces;
- UF Park-and-Ride Lot – 200 spaces;
- UF Hilton Convention Center Hotel – 200 spaces; and
- Rosa Parks Downtown Transfer Station – 10 spaces (coded more for its station-like amenities than for parking capacity).

5.3 Transit Access and Path-Building Model Results

Average weekday transit trips in the year 2007 were estimated by the RTS at 34,300. The model currently is estimating an average of 36,600 transit riders per day. This results in a deviation of less than seven percent. Additional details on transit assignment are provided in Section 8.0 of this report.

6.0 Mode Choice

Mode choice models can range from simple person-to-auto trip conversion models to more complex nested logit models that estimate modal shares among several categories of auto and transit modes. The Alachua County 2007 model uses a nested logit model approach for mode choice. This section of the report describes the structure and validation of the Alachua County 2007 mode choice model.

6.1 Mode Choice Model Structure

The standard FSUTMS process makes use of a nested logit model for mode choice, except in the case of 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 programs MATRIX and TRCOPY. This model uses the same mode choice constants that were used for the Alachua County 2000 model; however, mode choice coefficients and targets were carefully modified for a better match to mode splits for Gainesville and Alachua County, as reported in the latest data from the Census' American Community Survey (ACS).[8]

One script, MCMAT00A.S, directs the creation of a trip table containing five purposes: home-based work (HBW), home-based other (HBO), nonhome-based (NHB), home-based university (HBU) and UF Campus/Dorm (DORM) purposes. Transit fare data are compiled and restrained highway skims generated during trip distribution are input into the "MODE" module. The peak period utilizes the restrained skims, whereas the off-peak period uses the free flow skims. After running the mode choice model, the outputs are balanced into an origin and destination trip table. This trip table is then used during the highway and transit assignment phases of the model.

A separate script, MCMAT00C.S, combines trip purposes and outputs separate trip tables for the following modes:

1. Drive alone auto;
2. Carpool auto;
3. Light duty trucks;
4. Heavy duty trucks;
5. External-external trips;

[8] http://www.census.gov/acs/www/Products/index.html.

6. Peak period transit;
7. Off-peak period transit;
8. Nonmotorized 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 E, 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 Alachua County 2007 model was developed from the Alachua County 2000 model, with a few changes mainly to mode choice scripting (MCMAT00A.S). Validation efforts included iterative adjustment of parameter files, such as the coefficient (MCCOEFFICIENTS.CSV) and constant files (MCCONSTANTS.CSV). Care was taken to maintain coefficients in a numeric range consistent with current Florida DOT model validation standards, referenced earlier in this report and depicted in Table 6.1 below. As indicated, all 2007 coefficients for in-vehicle travel time (IVTT) are consistent with Florida DOT guidelines, which also reflect recent New Starts guidance from the Federal Transit Administration (FTA).

Table 6.1 Mode Choice Coefficients for 2007 Model

Mode Choice Model Parameters	2007 Alachua County	New FDOT Guidelines[a]
HBW IVTT[b]	-0.025	-0.02 to -0.03
HBNW IVTT (HBO IVTT)	-0.02	-0.002 to -0.01
NHB IVTT	-0.024	-0.02 to -0.03
UNI IVTT	-0.024	-0.02 to -0.03
HBW OVT[c]	-0.049	N/A
HBNW OVT (HBO OVT)	-0.048	N/A
NHB OVT	-0.07	N/A
UNI OVT	-0.048	N/A
HBW OVT/IVTT	2.0	2.0 to 3.0
HBNW OVT/IVTT (HBO OVT/IVTT)	2.4	2.0 to 3.0
NHB OVT/IVTT	2.9	2.0 to 3.0
UNI OVT/IVTT	2.0	2.0 to 3.0

[a] FSUTMS-Cube Framework Phase II: Model Calibration and Validation Standards Final Report.

[b] IVTT = in-vehicle travel time.

[c] OVTT = out-of-vehicle travel time.

6.3 Mode Choice Model Results

Mode choice results were adjusted to match statistics from Census ACS data for the home-based work (HBW) trip purpose. Also, home-based university (HBU) and UF Campus/Dorm (DORM) trip purposes were maintained to perform following suggestions provided by input from UF and RTS staff. With respect to transit mode split, Alachua County 2007 mode split percentages were greater than the 2000 model for the HBW purpose. The HBW purpose showed 4.39 percent of all HBW trips being transit as opposed to 0.75 percent for the Alachua County 2000 model and 3.38 percent from Census 2007 ACS. In total, 2.17 percent of all trips in the Alachua County 2007 model were allocated to transit modes while 2.35 percent of all trips in the Alachua County 2000 model were allocated to transit.

The total number of transit trips increased between the years 2000 and 2007, as a result of incentives put in place by RTS and UF to encourage transit ridership. While on the surface, it appears that HBU transit trips dropped between 2000 and 2007, this is primarily due to removing Later Gator and other Campus circulator bus routes from the year 2007 transit network, as mentioned earlier in Section 5.0. Any reductions in campus bound transit trips were shifted to nonmotorized travel modes (walk and bicycle), which were potentially underestimated in the 2000 model based on discussions with UF staff.

Validation runs with an even higher transit mode split resulted in substantial transit overassignment, as discussed later in this report. Since mode split targets are only available for the HBW purpose from Census ACS, a greater reliance was placed on matching unlinked transit ridership numbers by route, as provided by RTS. Transfer rates supplied by RTS were used to assess linked versus unlinked transit trips in an attempt to balance mode choice and transit assignment results.

Table 6.2 contains the Alachua County 2007 model Mode Choice validation results.

Table 6.2 Mode Choice Validation Summary

Trip Allocation By Mode	Alachua County 2007 Trips	Alachua County 2007 Percent of Trips	Alachua County 2000 Trips	Alachua County 2000 Percent of Trips	Census ACS[a] 2007 Classes	Census ACS[a] 2007 Percent of Trips
Home-Based Work						
Drive Alone	143,880	78.51%	122,692	81.72%	Drive Alone	76.78%
Two Passengers	15,951	8.70%	14,492	9.65%	Carpool	10.80%
Three+ Passengers	8,019	4.38%	7,249	4.83%		
Total Transit	8,043	4.39%	1,125	0.75%	Transit	3.38%
Walk	4,098	2.24%	2,947	1.96%	Walk	3.39%
Bike	3,278	1.79%	1,641	1.09%	Motor/Bicycle	3.54%
HBW Total	**183,269**	**100.00%**	**150,146**	**100.00%**		
Home-Based Other						
Drive Alone	200,318	39.63%	167,037	38.10%		
Two Passengers	191,683	37.92%	169,427	38.64%		
Three+ Passengers	90,914	17.99%	83,985	19.15%		
Total Transit	2,943	0.58%	1,375	0.31%		
Walk	17,952	3.55%	15,134	3.45%		
Bike	1,631	0.32%	1,497	0.34%		
Total	**505,441**	**100.00%**	**438,455**	**100.00%**		
Nonhome-Based						
Drive Alone	166,902	50.70%	129,768	45.29%		
Two Passengers	106,625	32.39%	97,798	34.13%		
Three+ Passengers	45,341	13.77%	47,888	16.71%		
Total Transit	2,587	0.79%	2,033	0.71%		
Walk	5,105	1.55%	6,085	2.12%		
Bike	2,640	0.80%	2,981	1.04%		
Total	**329,200**	**100.00%**	**286,553**	**100.00%**		
Home-Based University						
Drive Alone	34,900	45.96%	26,274	49.75%		
One Passenger	4,399	5.79%	3,098	5.87%		
Two+ Passenger	2,261	2.98%	1,555	2.94%		
Total Transit	9,178	12.09%	11,303	21.40%		
Walk	15,025	19.78%	5,546	10.50%		
Bike	10,179	13.40%	5,032	9.53%		
Total	**75,942**	**100.00%**	**52,808**	**100.00%**		
UF Campus/Dorm						
Total Transit	1,981	8.40%	6,624	25.00%		
Walk	15,615	66.25%	12,810	48.36%		
Bike	5,974	25.35%	7,057	26.64%		
Total	**23,570**	**100.00%**	**26,491**	**100.00%**		
All Purposes						
Drive Alone	546,000	47.80%	445,771	46.70%		
One Passenger	318,658	27.90%	284,815	29.84%		
Two+ Passenger	146,535	12.83%	140,677	14.74%		
Total Transit	24,732	2.17%	22,460	2.35%		
Walk	57,795	5.06%	42,522	4.46%		
Bike	23,702	2.08%	18,208	1.91%		
Total	**1,142,154**	**100.00%**	**954,453**	**100.00%**		

[a] Source: 2007 Census ACS Data for Alachua County.

7.0 Highway Assignment

The purpose of a highway assignment model is to load auto trips onto a highway network, resulting in traffic estimates on individual links that ultimately attempt to simulate general vehicular travel patterns throughout the study area. For the Alachua County 2007 model, a series of postprocessing steps also are accomplished during highway assignment to generate output statistics, create new network attributes, and add nonmotorized trips to the highway network as a separate loaded "purpose".

Validation of the highway assignment involved adjustments to external travel and trip generation assumptions, iterative highway network modifications, adjustment of model speeds, and other changes related to the transit system to shift trips among modes most effectively. A number of key evaluation statistics were generated during the assignment phase of the model. Volume-over-count ratios were compared by area type, facility type, laneage, and screenline and volume groups. Along with these statistics, the root mean square error (RMSE) was generated and evaluated by volume group.

This section describes validation of the highway assignment model. It includes an overview of the model structure, development, and iterative adjustment of model inputs and parameters, and a review of final model validation results.

■ 7.1 Highway Assignment Model Structure

Auto trips are loaded onto the network by means of an iterative equilibrium highway load program based on an all or nothing capacity restrained assignment algorithm. A series of statistical summaries are subsequently generated through postprocessing steps. The most significant statistics for highway assignment validation are generated in reports collectively known as "Highway Evaluation or HEVAL" in conjunction with RMSE. Postprocessing scripts from the 2000 model were modified during the 2007 validation to produce additional assignment statistics and summaries not available in the previous model. The scripts and corresponding model flowchart for highway assignment can be found in Appendix E.

■ 7.2 Development and Validation of Highway Assignment Model

In total, 23 model runs were executed in order to validate the Alachua County 2000 model, excluding numerous test runs that were necessary for iterative validation adjustments.

Model validation was accomplished by minimizing the difference between model estimated volumes and observed traffic counts for the year 2007 on network links throughout the study area. As many count locations were accounted for as possible in order to ensure a wide range of coverage geographically as well as to incorporate as many examples of facilities and land uses located within the study area.

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 by the Consultant in order to identify areas for improvement in the model and successful strategies toward validation enhancement. Appropriate changes consistent with the findings revealed during analysis of results were then implemented and subsequent runs were executed. This iterative process was continued until validation accuracy standards were achieved.

Changes made to the model during highway assignment validation consisted mainly of iterative adjustments to speeds and highway network editing, including adjustment of centroids and centroid connectors, verifying highway alignments against street GIS layers and aerial photography, and modifying highway characteristics based on input and comments that were provided by the Metropolitan Transportation Planning Organization for the Gainesville Urbanized Area Long Range Transportation Plan Technical Advisory Committee Subcommittee.

■ 7.3 Highway Assignment Validation Results

Perhaps the most common use of travel demand models is to forecast future traffic volumes in order to identify the impacts of growth over time and better plan to mitigate these impacts. In the Gainesville Urbanized Area, a greater focus is placed on transit mobility than most comparably sized areas and validation of the Alachua County 2007 model likewise focused more effort on transit assignment accuracy. Even under these conditions, however, proper validation of the highway assignment is critical to the meaningful use of travel demand models as the highway network forms the backbone of the transit network and is assumptions. Key statistics analyzed as part of the validation process include the following:

- Volume-over-count ratios on count locations;
- Percent root mean square error;
- Volume-over-count ratios along screenlines;
- Vehicle-miles traveled (VMT); and
- Vehicle-hours traveled (VHT).

Each of the above measures is discussed separately in the remainder of this section.

Assignment Performance by Area Type/Facility Type/Lanes Categories

The areawide accuracy of highway assignment is measured, in part, by means of volume-over-count ratios for area type, facility type, and lanes categories. FDOT standards generally allow for an accuracy of +/- 15 percent per category and +/- five percent areawide. The Alachua County 2007 model achieves the areawide accuracy for volume-over-count ratio at 1.01. There were no occurrences of link group volume-over-count ratios (i.e., area type and facility type categories) that exceeded the standard tolerances by group.

Table 7.1 demonstrates a detailed record of the volume-over-count ratios for each link group category of area type, facility type, and number of lanes. In comparing accuracy of the 2007 model against the 2000 model, the results are comparable. At the areawide level, both models show a percent error of two percent, the difference is that the 2000 base year highway assignment was underestimating vehicle trips by two percent whereas the 2007 base year model is overestimating these trips by two percent.

In looking at validation by area type, facility type, and laneage categories, there are examples where one model is better than the other but nothing particularly troublesome. Some key validation improvements over the 2000 model include undivided arterials, collectors, and central business districts (CBD). Ramp counts were not included in the 2000 model and validation of these affects validation of the mainline I-75 corridor. Achieving a higher volume-over-count ratio on freeways also would mean a higher ratio on ramps, which are presently running as high as one would want at present (+14 percent).

Table 7.1 Volume to Count Performance by Category

		Volumes Over Counts	
		Percent Difference (+/- 15%)	
		Alachua County	
Category		**2007**	**2000**
Facility Type	Freeway	0.94	1.01
	Divided Arterial	1.08	1.02
	Undivided Arterial	0.99	0.92
	Collectors	0.89	0.79
	One-Way/Frontage	0.93	1.00
	Ramp	1.14	N/A
Area Type	CBD	1.05	0.89
	CBD Fringe	0.98	1.01
	Residential	0.92	0.93
	OBD	1.06	N/A
	Rural	1.10	1.01

Number of Lanes	One Lane	0.95	N/A
	Two Lanes	1.03	N/A
	Three Lanes	1.08	N/A
Total		**1.02**	**0.98**

Screenline Performance

Analyzing volume-over-count ratios along screenlines allows for examining flows into, out of, and across geographic subareas and corridors. This constitutes a key component of highway assignment as well as assisting in the examination of trip distribution patterns. There are 10 screenlines in the Alachua County 2007 model. An external cordon measuring trips coming into and going out of the study area is included as well. Figure 7.1 depict the screenlines used in the Alachua County 2007 model.

FDOT has established four ranges for measuring accuracy based on total counts comprising each screenline. Screenlines that carry less than 35,000 vehicles per day (VPD) should validate within +/- 20 percent. Screenlines that carry between 35,000 to 70,000 VPD should validate within +/- 15 percent. Screenlines that carry more than 70,000 VPD should validate within +/- 10 percent. External cordons should validate within +/- one percent. Out of 10 total screenlines, all met established accuracy targets. The results of the volume-over-count ratios by screenline are depicted in Table 7.2.

Percent Root Mean Square Error

The percent root mean square error (RMSE) indicates whether the simulated network contains an acceptable level of assignment error. This is based on both areawide and volume group summaries. Accuracy is more stringent for higher volume facilities than for lower volume facilities, as the same percent error equals a higher assignment volume.

No RMSE category failed to meet established accuracy ranges with the Alachua County 2007 model. The overall RMSE for the study area was 31.7 percent, below the 32 to 39 percent minimum recommended accuracy range for areawide total RMSE. This is better than the Alachua County 2000 model, which reported a total RMSE of 32.8 percent. These results are summarized in Table 7.3. RMSEs are significantly improved in the Alachua County 2007 model versus the Alachua County 2000 model especially for the volume group with less than 5,000 VPD by 9.6 percent and for the groups between 30,000 and 50,000 VPD by 7.2 percent.

Streets with less than 5,000 VPD are mostly urban collectors and rural arterials so this improved accuracy equates with a better estimate of trips coming into the Gainesville urbanized area from outlying areas as well as neighborhood collectors and circulators. Urban arterials with 30,000+ VPD are among the most important corridors in the Gainesville urbanized area and an enhanced accuracy here means a greater confidence in proposals for major investments, regardless of mode.

Figure 7.1 Gainesville MTPO 2007 Model Screenlines

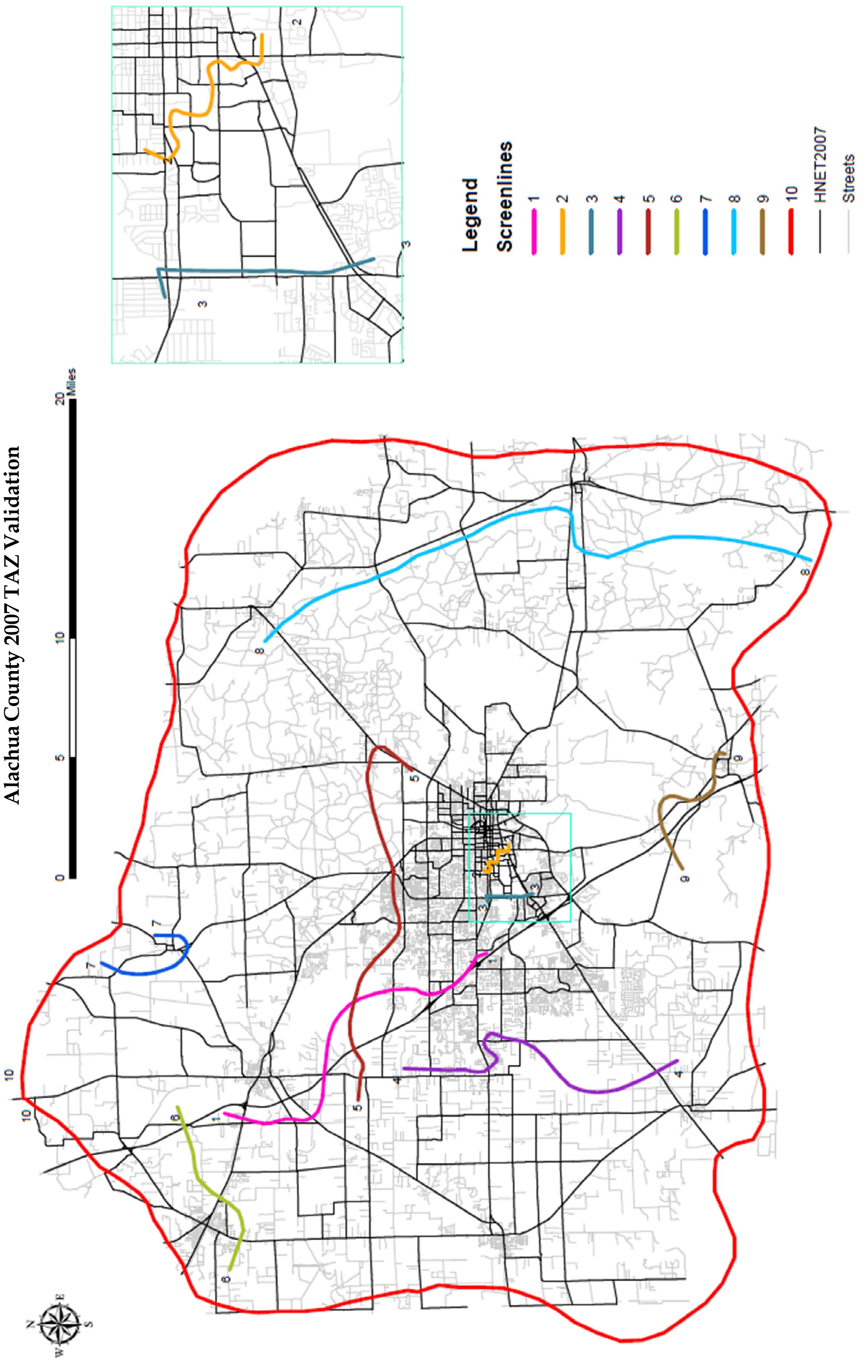


Table 7.2 Volume to Count Performance by Screenline

	Alachua County		V/C Ratio		
	2007		2007		
Screenline	**Total Volume**	**Total Count**	**Alachua County**	**FDOT Accuracy Level[a]**	**Screenline Descriptions**
1	152,030	145,068	1.05	+/- 10%	1 Crossing I-75
2	141,566	133,473	1.06	+/- 10%	2 Crossing East UF Campus
3	153,847	133,655	1.15	+/- 10%	3 Crossing SR 121
4	66,168	62,873	1.05	+/- 15%	4 EW Cutline west of I-75
5	148,081	141,938	1.04	+/- 10%	5 NS Crossing SR-222 (39th Avenue)
6	86,335	82,100	1.05	+/- 10%	6 NS Cutline in NW County/High Springs
7	11,255	10,619	1.06	+/- 20%	7 La Crosse Area
8	35,479	37,126	0.96	+/- 15%	8 EW Crossing U.S. 301
9	92,341	90,827	1.02	+/- 10%	9 Micanopy Area
10	244,482	244,474	1.00	+/- 1%	10 External Cordon

[a] *FSUTMS-Cube Framework Phase II: Model Calibration and Validation Standards Final Report,* October 2008.

Table 7.3 Root Mean Squared Error

Count Range	**Accuracy Range[a]**	**2007 Alachua County**	**1990 Alachua County**
1-5,000	45-55	46.0%	55.6%
5,000-10,000	35-45	30.7%	30.2%
10,000-20,000	25-35	25.4%	22.2%
20,000-30,000	15-27	21.4%	15.4%
30,000-40,000	22-24	8.3%	25.8%
40,000-50,000	20-22	18.6%	N/A
Average Total	**32-39**	**31.7%**	**32.8%**

[a] FSUTMS-Cube Framework Phase II: Model Calibration and Validation Standards Final Report, October 2008.

8.0 Transit Assignment

The transit assignment step in the Alachua County 2007 model loads trips to the transit network. Separate loads are conducted by mode and period as allocated in the mode choice model. HBW trips are assigned to the "peak period" network while HBNW and NHB trips are assigned to the "midday" network (so in reality this is more of a comparison between work and nonwork trips than time periods). 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 daily "unlinked" route ridership data provided by the RTS.

The Alachua County 2007 transit assignment model estimated approximately 36,599 "unlinked" riders systemwide, while the RTS reported 34,326 daily ridership systemwide for FY 2007 as shown in Table 8.1. For comparison purposes, only those bus routes that were included in the Alachua County 2007 model were reported in Table 8.1. On average, the ridership for the Alachua County 2007 model differs by only 6.6 percent from data provided by the RTS. This is within the preferred 15 percent FDOT validation standard.

On a route-by-route basis, the Alachua County 2007 transit assignment model validates well with 23 out of 29 bus routes performing within the preferred accuracy range. Of the remaining bus routes, four bus routes are performing within the acceptable range (but not the "preferred" range) as well. Therefore, 27 out of 29 bus routes are assigning within FDOT standards. In general, four-step models do not typically validate well on a segment-by-segment basis for local bus routes. Other applications do exist that are more appropriate for use in assessing local bus operational adjustments. However, through validation efforts and iterative modification of input and parameter files, the Alachua County 2007 model performs well in transit assignment not only as a system but on a route-by-route basis as well.

At the bottom of Table 8.1, are a number of interrelated statistics such as linked trips, unlinked trips, transfers, and annual ridership. "Observed" linked trips were calculated by applying a 16 percent transfer rate documented in the recent on-board rider survey. "Estimated" linked trips are the number of transit trips output by the mode choice model. Even though the linked trips from mode choice are somewhat lower than desired, it was felt that the observed unlinked ridership numbers were more defensible (derived from passenger counts) than the linked observed number (derived from on-board survey response to a question about transfers). Any increase in the number of linked trips would always result in more unlinked trips as well, and systemwide model estimates of unlinked trips are very close to RTS observed numbers.

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

Table 8.1 Year 2007 Transit Loading Estimates

Route	Route Description	RTS Ridership 2007	Alachua County 2007 Model	Percent Error	Acceptable (+/-)[a]	Preferable (+/-)[b]
Route 1	Butler Plaza to Downtown via Archer Road	1,667	2,768	66.1%	100%	65%
Route 2	Downtown to Robinson Heights via SE 15th Street	343	31	-91.0%	150%	100%
Route 5	Oaks Mall to Downtown via University Avenue	1,562	2,291	46.7%	100%	65%
Route 6	Downtown to Gainesville Mall via 6th Avenue	361	307	-14.9%	150%	100%
Route 7	Downtown to Eastwood Meadows	406	41	-89.9%	150%	100%
Route 8	Pine Ridge to Shands via NW 13th Street	1,199	2,288	90.8%	100%	65%
Route 9	Lexington Crossing to McCarty Hall	2,764	2,066	-25.3%	65%	35%
Route 10	SFCC to Downtown via NW 16th Avenue/University Avenue	280	149	-46.8%	150%	100%
Route 11	Eastwood Meadows to Downtown via University Avenue	411	253	-38.4%	150%	100%
Route 12	Campus Club to McCarty Hall	2,430	3,454	42.1%	65%	35%
Route 13	Job Services to Newell Drive/Museum Road via 13th Street	1,488	840	-43.6%	100%	65%
Route 15	Downtown to NW 23rd Street/NW 6th Street	998	564	-43.5%	150%	100%
Route 16	Newell Drive/Museum Road to Sugar Hill via 16th Avenue	1,116	991	-11.2%	100%	65%
Route 20	Oaks Mall to McCarty Hall via SW 20th Avenue	3,122	2,520	-19.3%	65%	35%
Route 21	SW 43rd Street to McCarty Hall	1,567	1,574	0.4%	100%	65%
Route 24	Downtown to Job Corps via SR 24 (Waldo Road)	386	257	-33.3%	150%	100%
Route 34	Lexington Crossing to the Hub	1,366	758	-44.5%	100%	65%
Route 35	McCarty Hall to Homestead Apartments	2,113	2,190	3.6%	65%	35%
Route 36	McCarty Hall to SW 34th Street/Archer Road	731	1,790	145.0%	150%	100%
Route 43	SFCC to Downtown via 43rd Street	603	1,579	162.0%	150%	100%
Route 75	Butler Plaza to Oaks Mall via 75th Street	815	247	-69.7%	150%	100%
Route 117	Park-and-Ride 2 (SW 34th Street)	830	1,433	72.7%	150%	100%
Route 118	Park-and-Ride 1 (Harn Museum)	3,070	2,282	-25.60%	65%	35%
Route 119	Family Housing	346	294	-14.90%	150%	100%

Table 8.1 Year 2007 Transit Loading Estimates (continued)

Route	Route Description	RTS Ridership 2007	Alachua County 2007 Model	Percent Error	Acceptable (+/-)[a]	Preferable (+/-)[b]
Route 120	West Circulator (Fraternity Row)	1,198	1,158	-3.30%	100%	65%
Route 121	Commuter Lot	1,020	2,061	102.10%	100%	65%
Route 122	UF North/South Circulator	220	369	67.70%	150%	100%
Route 125	Lakeside	955	1,006	5.30%	150%	100%
Route 127	East Circulator (Sorority Row)	962	1,039	8.00%	150%	100%
Route 120, 122, 127	All UF Circulators	2,379	2,565	7.80%	65%	35%
Total Unlinked Riders		34,326	36,599	6.60%	+/- 20 %	+/- 15 %
Without Circulators		31,947	34,034	6.50%		
Transfers*		5,492				
Total Linked Trips		28,834	24,732	-14.20%	+/- 9 %	+/- 3 %
FY 07 Annual Riders		8,939,334				
Expansion Factor		252.28				

[a] Based on 16 percent reported in RTS on-board survey.

[b] FSUTMS-Cube Framework Phase II: Model Calibration and Validation Standards, Final Report, October 2008.

9.0 2035 E+C Scenario

Section 9.0 discusses development of the 2035 Existing-plus-Committed (E+C) scenario. Technical Report No. 3 of this series describes the development of the E+C networks and 2035 external trips. Year 2035 socioeconomic data and external trips were combined with E+C highway and transit projects to generate forecasts of highway and transit trips. Iterative adjusting of bus fares showed that the model was overly sensitive to fare changes. Discussions with RTS staff led to a decision on implementation of a Bus Fare Factor.

The Bus Fare Factor was introduced as a Scenario Key that can be edited by model user should alternative fare changes be tested using the model. Bus fare factor (BUSFAREFAC) is a dollar amount. The base year 2007 local bus fare is $1.00, while the current bus fare has increased to $1.50 as of the year 2010, when the model was validated. Therefore, any future scenarios, including the E+C scenario, should use a bus fare of no less than $1.50.

Initial efforts to change the fare from $1.00 to $1.50 in the bus fare text input file (ALACHUA.FAR) were not successful, resulting in severely reduced transit trips and ridership for future year scenarios. Due to this reason, a Bus Fare Factor (BUSFAREFAC) was introduced so that the transit mode choice script minimizes the impact from the transit fare increase to 10 percent, consistent with ridership changes experienced and documented by the RTS. Details on how the mode choice script was modified can be found in Appendix E, along with the model process flowchart.

10.0 Summary and Conclusions

The model validation phase of the Gainesville Urbanized Area Year 2035 Long Range Transportation Plan (LRTP) Update essentially began with data development and review which was documented in Technical Report 3. Data review, adjustment, and correction was an iterative process throughout the model validation effort, reflecting identification of data issues based on model results.

Once the input data were initially deemed sufficient to proceed, work continued on validating each component of the FSUTMS travel demand model. A validation worksheet was prepared and used to summarize each validation run and model performance against prior versions of the Alachua County model as well as other comparable travel demand models. This updated validation worksheet incorporates the latest FDOT model validation accuracy standards and allows for a complete record of all model adjustments made during the validation effort and the resulting impacts.

Model results were shared periodically with staff from the MTPO, FDOT District Two, Alachua County, UF staff, RTS, and other members of the Metropolitan Transportation Planning Organization (MTPO) for the Gainesville Urbanized Area LRTP Technical Advisory Committee Subcommittee. 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 aforementioned Subcommittee agreed to focus much of the validation effort on the transit side of the model, as the MTPO Board has demonstrated a strong commitment to the role of transit in the area's future mobility.

The base year 2007 Alachua County travel demand forecasting model meets most established FSUTMS and national standards for model accuracy and reasonableness. The validated model has been used in subsequent phases of the Gainesville Urbanized Area Year 2035 LRTP Update to develop and test numerous transportation needs alternatives, as described in other Technical Reports of this series.

Appendix A: Socioeconomic Data Format

■ ZONEDATA{YEAR}.DBF

Notes: ZONEDATA{YEAR}.DBF in the Gainesville MTPO 2007 represents and combines the previous 2000 MTPO model input files of ZDATA1, ZDATA2, UF Data, PEV (Pedestrian Environment Variable) files.

Please also note that ZONEDATA{YEAR}.DBF is part of GIS TAZ data of ZONEDATA.SHP, together with ZONEDATA.SHX. When editing this ZoneData file, Cube or GIS software must be used.

Attribute List for Population and Household Data

TAZ_2007 – Traffic Analysis Zone (TAZ) number in the Gainesville MTPO 2007 model. Future year ZoneData35.DBF also uses this attribute for TAZs.

SFDU – Number of single-family dwellings units

SPOP – Population in single-family dwellings units

MFDU – Number of multifamily dwellings units (MFDUS)

MPOP – Population in MFDUS

TOTPOP07 – Total population for year 2007 (this attribute is not used by model scripts, instead SPOP and MPOP 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(referred to as the ZDATA2 file in 2000 model)

OIEMP – Other industrial employment

MFGEMP – Manufacturing industrial employment

COMEMP – Commercial employment

SERVEMP – Service employment

TOTEMP – Total employment

SCHENR – School enrollment by school location (this excludes any UF or Santa Fe College enrollment)

Attribute List for University of Florida Data

UF_EMP – Number of UF place-of-work employees by TAZ (this variable also is used to reallocate service employment on UF Campus)

UF_DORM_ST – Number of on-campus UF student residents

UF_PARKING – UF commuting parking spaces, excluding on-campus student long-term not

used for commuting (this variable also is used to reallocate service employment on UF Campus)

CLASSROOMS – Number of UF classrooms (model scripts do not directly use this)

CLASSSQFT – Square feet of UF classrooms (model scripts do not directly use this)

SEATS – Number of UF classroom seats

UF-OC-ST – Number of UF off-campus student residents, estimated from student address records

provided by UF

SUB_AREA – Name of city or incorporated area or Alachua if a zone is within the unincorporated area

UFZONES – Identifier that indicates that a zone is on UF 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 – Nonmotorized 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 by model users)

COMPOSIT – composite PEV value (model scripts do not directly 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 of SELZONE_MOTOR in final highway assignment output network of COMBINEDLOADED.NET)

HOTEL – Identifier used in the previous model (model scripts do not directly use this)

Attribute List for Parking Data (previously found in ZDATA2 file in 2000 model)

SHORTPARK – Short-term (3 hour) parking cost (cents)

LONGPARK – Long-term (8 hour) parking cost (cents)

STUDENTPAR – Student (8 hour) parking cost (cents) at UF

Attribute List for Population and Household Variable Data (previously found in ZDATA1 file in 2000 model)

SF_SEA – Percent SFDU not occupied by permanent residents

SF_0V – Percent households having no vehicles in SFDU occupied by permanent residents

SF_1V – Percent households having one vehicle in SFDU occupied by permanent residents

SF_2V – Percent households having two vehicles in SFDU occupied by permanent residents

SF_3V – Percent households having three or more vehicles in SFDU occupied by permanent residents

SF_VAC – Percent SFDU vacant

MF_SEA – Percent MFDU not occupied by permanent residents

MF_0V – Percent households having no vehicles in MFDU occupied by permanent residents

MF_1V – Percent households having one vehicle in MFDU occupied by permanent residents

MF_2V – Percent households having two vehicles in MFDU occupied by permanent residents

MF_3V – Percent households having three or more vehicles in MFDU occupied by permanent residents

MF_VAC – Percent MFDU vacant

HM_POC – Percent hotel-motel units occupied

Appendix B: Speed Capacity Adjustments and VFACTORS

Table B.1 Speed Capacity Adjustments

LOW AREA TYPE	HIGH AREA TYPE	LOW FACILITY TYPE	HIGH FACILITY TYPE	LOW LANES	HIGH LANES	CAPACITY OPERATOR	CAPACITY	SPEED OPERATOR	SPEED
10	19	21	29	1	9		1	*	1.50
10	19	30	99	1	9		1	*	1.25
10	19	60	64	1	9		1	*	0.80
10	59	40	49	1	9		1	*	1.20
20	39	11	12	1	9		1	+	13.00
31	31	25	25	1	8		1	*	1.20
40	43	11	12	1	9		1	+	11.00
50	52	11	12	1	9		1	+	10.00

Table B.2 Variable Factors

FACILITY TYPE	(UROAD) PRACTICAL / ABSOLUTE CAPACITY RATIO	(CONFAC) PEAK-to-DAILY CAPACITY FACTOR	BPR (α) LEVEL-of-SERVICE VALUE	BPR (β) EXPONENT
10	0.68	0.1	0.15	6.5
11	0.68	0.09	0.15	6.5
12	0.68	0.09	0.15	6.5
13	1	0.1	0.15	6.5
14	1	0.1	0.15	6.5
15	0.68	0.1	0.15	6.5
16	0.68	0.1	0.15	6.5
17	0.68	0.1	0.15	6.5
18	1	0.1	0.15	6.5
19	0.68	0.1	0.15	6.5
20	0.92	0.1	0.15	5.5
21	0.73	0.1	0.15	5.5
22	0.73	0.1	0.15	5.5
23	0.81	0.1	0.15	5.5
24	0.95	0.1	0.15	5.5
25	0.96	0.1	0.15	5.5
26	1	0.1	0.15	5.5
27	1	0.1	0.15	5.5
28	1	0.1	0.15	5.5
29	1	0.1	0.15	5.5
30	0.92	0.1	0.15	4.5
31	0.68	0.1	0.15	4.5
32	0.81	0.1	0.15	4.5
33	0.95	0.1	0.15	4.5
34	0.88	0.1	0.15	4.5
35	0.68	0.1	0.15	4.5
36	0.81	0.1	0.15	4.5
37	0.95	0.1	0.15	4.5
38	0.96	0.1	0.15	4.5
39	1	0.1	0.15	4.5
40	0.86	0.1	0.15	4.5
41	0.92	0.1	0.15	4.5
42	0.92	0.1	0.15	4.5
43	0.92	0.1	0.15	4.5

44	0.86	0.1	0.15	4.5
45	0.86	0.1	0.15	4.5
46	0.86	0.1	0.15	4.5
47	0.86	0.1	0.15	4.5
48	0.86	0.1	0.15	4.5
49	1	0.1	0.15	4.5
50	1	0.1	0.15	4.5
51	1	0.1	0.15	4.5
52	1	0.1	0.15	4.5
53	1	0.1	0.15	4.5
54	1	0.1	0.15	4.5
55	1	0.1	0.15	4.5
56	1	0.1	0.15	4.5
57	1	0.1	0.15	4.5
58	1	0.1	0.15	4.5
59	1	0.1	0.15	4.5
60	0.96	0.1	0.15	4.5
61	0.68	0.1	0.15	4.5
62	0.81	0.1	0.15	4.5
63	0.95	0.1	0.15	4.5
64	0.96	0.1	0.15	4.5
65	0.68	0.1	0.15	4.5
66	0.81	0.1	0.15	4.5
67	0.95	0.1	0.15	4.5
68	0.96	0.1	0.15	4.5
69	1	0.1	0.15	4.5
70	0.68	0.1	0.15	6.5
71	0.51	0.1	0.15	6.5
72	0.92	0.1	0.15	6.5
73	0.51	0.1	0.15	6.5
74	0.92	0.1	0.15	6.5
75	0.51	0.1	0.15	6.5
76	0.92	0.1	0.15	6.5
77	0.51	0.1	0.15	6.5
78	0.92	0.1	0.15	6.5
79	0.68	0.09	0.15	6.5
80	0.68	0.1	0.3	8.5
81	0.68	0.1	0.3	8.5
82	0.68	0.1	0.3	8.5
83	0.68	0.1	0.3	8.5
84	0.68	0.1	0.3	8.5

85	0.68	0.1	0.3	8.5
86	0.68	0.1	0.3	8.5
87	0.68	0.1	0.3	8.5
88	0.68	0.1	0.3	8.5
89	0.68	0.1	0.3	8.5
90	0.68	0.1	0.15	6.5
91	0.68	0.1	0.15	6.5
92	0.68	0.1	0.15	6.5
93	0.68	0.1	0.15	6.5
94	0.68	0.1	0.15	5.5
95	0.68	0.1	0.15	4.5
96	1	0.1	0.15	6.5
97	0.51	0.1	0.15	6.5
98	0.51	0.1	0.15	6.5
99	1	0.1	0.15	6.5

Appendix C: 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
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
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
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 D: Friction Factors

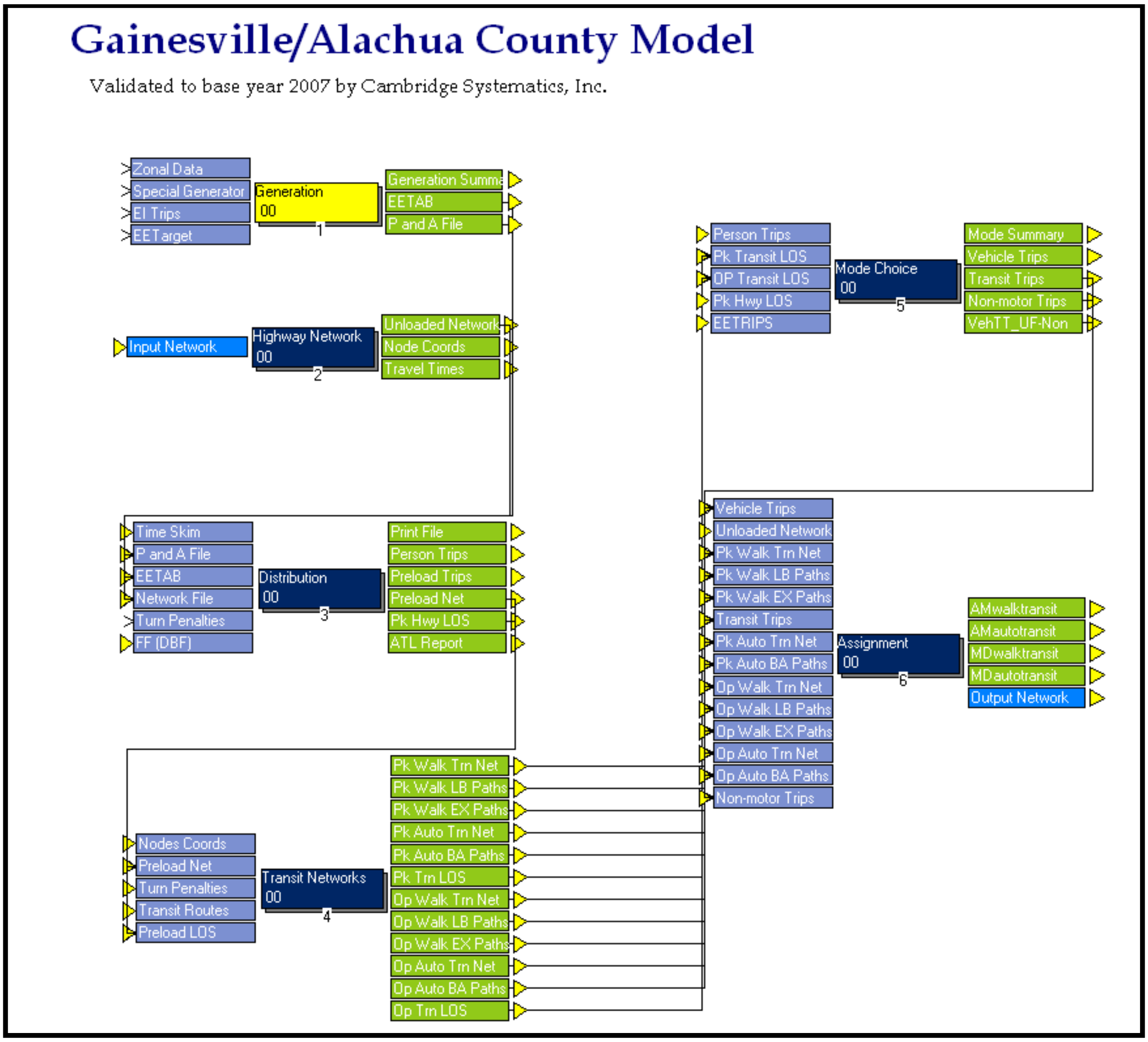
TIME	HBWFF	HBSHFF	HBSRFF	HBOFF	NHBFF	TK4FF	TKSGLFF	TKTRLRFF	SOVIEFF	HOVIEFF	TKLTIEFF	TKHTIEFF	HBUFF	HDORMUFF
1	25,208	12,6687	12,6687	12,6687	19,8262	9,231	9,048	9,704	222	222	222	222	126,687	126,687
2	21,983	47,324	47,324	47,324	71,259	8,521	8,187	9,418	333	333	333	333	47,324	47,324
3	19,282	25,585	25,585	25,585	37,571	7,866	7,408	9,139	444	444	444	444	25,585	25,585
4	16,953	16,092	16,092	16,092	23,174	7,261	6,703	8,869	555	555	555	555	16,092	16,092
5	14,924	10,997	10,997	10,997	15,577	6,703	6,065	8,607	666	666	666	666	10,997	10,997
6	13,149	7,919	7,919	7,919	11,056	6,188	5,488	8,353	777	777	777	777	7,919	7,919
7	11,591	5,913	5,913	5,913	8,147	5,712	4,966	8,106	888	888	888	888	5,913	5,913
8	10,222	4,534	4,534	4,534	6,170	5,273	4,493	7,866	1,333	1,333	1,333	1,333	4,534	4,534
9	9,018	3,548	3,548	3,548	4,773	4,868	4,066	7,634	1,666	1,666	1,666	1,666	3,548	3,548
10	7,957	2,820	2,820	2,820	3,753	4,493	3,679	7,408	3,333	3,333	3,333	3,333	2,820	2,820
11	7,023	2,271	2,271	2,271	2,991	4,148	3,329	7,189	6,666	6,666	6,666	6,666	2,271	2,271
12	6,199	1,849	1,849	1,849	2,410	3,829	3,012	6,977	7,777	7,777	7,777	7,777	1,849	1,849
13	5,473	1,519	1,519	1,519	1,960	3,535	2,725	6,771	8,888	8,888	8,888	8,888	1,519	1,519
14	4,833	1,257	1,257	1,257	1,607	3,263	2,466	6,570	9,999	9,999	9,999	9,999	1,257	1,257
15	4,267	1,047	1,047	1,047	1,326	3,012	2,231	6,376	9,999	9,999	9,999	9,999	1,047	1,047
16	3,769	877	877	877	1,101	2,780	2,019	6,188	9,999	9,999	9,999	9,999	877	877
17	3,328	739	739	739	919	2,567	1,827	6,005	9,999	9,999	9,999	9,999	739	739
18	2,940	625	625	625	771	2,369	1,653	5,827	9,999	9,999	9,999	9,999	625	625
19	2,597	531	531	531	649	2,187	1,496	5,655	9,999	9999	9,999	9,999	531	531
20	2,294	452	452	452	548	2,019	1,353	5,488	6,666	6,666	6,666	6,666	452	452
21	2,026	387	387	387	465	1,864	1,225	5,326	3,333	3,333	3,333	3,333	387	387
22	1,790	331	331	331	395	1,720	1,108	5,169	1,111	1,111	1,111	1,111	331	331
23	1,582	285	285	285	337	1,588	1,003	5,016	444	444	444	444	285	285
24	1,397	246	246	246	288	1,466	907	4,868	222	222	222	222	246	246
25	1,235	212	212	212	247	1,353	821	4,724	111	111	111	111	212	212
26	1,091	184	184	184	212	1,249	743	4,584	66	66	66	66	184	184
27	964	159	159	159	183	1,153	672	4,449	22	22	22	22	159	159
28	852	138	138	138	157	1,065	608	4,317	16	16	16	16	138	138
29	753	120	120	120	136	983	550	4,190	13	13	13	13	120	120
30	665	105	105	105	118	907	498	4,066	11	11	11	11	105	105
31	588	92	92	92	102	837	450	3,946	16	16	16	16	92	92

TIME	HBWFF	HBSHFF	HBSRFF	HBOFF	NHBFF	TK4FF	TKSGLFF	TKTRLRFF	SOVIEFF	HOVIEFF	TKLTIEFF	TKHTIEFF	HBUFF	HDORMUFF
32	519	80	80	80	88	773	408	3,829	3	3	3	3	80	80
33	459	70	70	70	77	714	369	3,716	1	1	1	1	70	70
34	406	61	61	61	67	659	334	3,606	1	1	1	1	61	61
35	358	54	54	54	58	608	302	3,499	1	1	1	1	54	54
36	317	47	47	47	51	561	273	3,396	1	1	1	1	47	47
37	280	41	41	41	44	518	247	3,296	1	1	1	1	41	41
38	247	36	36	36	39	478	224	3,198	1	1	1	1	36	36
39	219	32	32	32	34	442	202	3,104	1	1	1	1	32	32
40	193	28	28	28	29	408	183	3,012	1	1	1	1	28	28
41	171	25	25	25	26	376	166	2,923	1	1	1	1	25	25
42	151	22	22	22	23	347	150	2,837	1	1	1	1	22	22
43	133	19	19	19	20	321	136	2,753	1	1	1	1	19	19
44	118	17	17	17	17	296	123	2,671	1	1	1	1	17	17
45	104	15	15	15	15	273	111	2,592	1	1	1	1	15	15
46	92	13	13	13	13	252	101	2,516	1	1	1	1	13	13
47	81	12	12	12	12	233	91	2,441	1	1	1	1	12	12
48	72	11	11	11	10	215	82	2,369	1	1	1	1	11	11
49	64	9	9	9	9	198	74	2,299	1	1	1	1	9	9
50	56	8	8	8	8	183	67	2,231	1	1	1	1	8	8
51	50	7	7	7	7	169	61	2,165	1	1	1	1	7	7
52	44	7	7	7	6	156	55	2,101	1	1	1	1	7	7
53	39	6	6	6	6	144	50	2,039	1	1	1	1	6	6
54	34	5	5	5	5	133	45	1,979	1	1	1	1	5	5
55	30	5	5	5	4	123	41	1,920	1	1	1	1	5	5
56	27	4	4	4	4	113	37	1,864	1	1	1	1	4	4
57	24	4	4	4	3	105	33	1,809	1	1	1	1	4	4
58	21	3	3	3	3	97	30	1,755	1	1	1	1	3	3
59	19	3	3	3	3	89	27	1,703	1	1	1	1	3	3
60	16	3	3	3	2	82	25	1,653	1	1	1	1	3	3
120	0	0	0	0	0	0	0	0	0	0	0	0	0	0

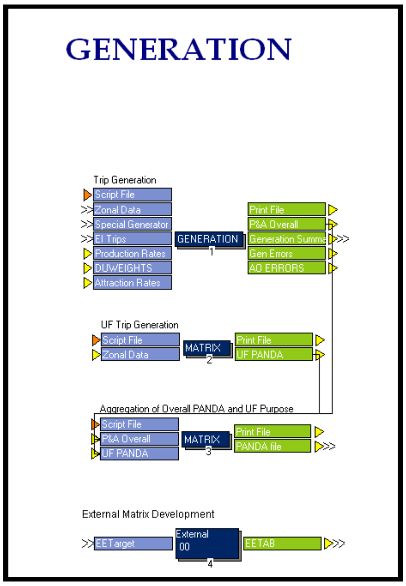
Appendix E: Model Flowchart, Scripts and File Locations

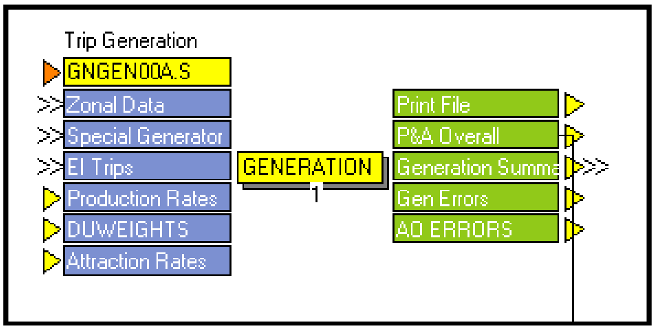
Gainesville 2007 Base Year Model Architecture
File Names, Locations, and Sources

Model Step	File Name	File Format/ Extension	File Type	Folder Location	Initial Source
Trip Generation	ATTRRATES	DBF	Parameters	\Parameters	Olympus
	DUWEIGHTS	DBF	Parameters	\Parameters	Census 2000
	EETARGETS	DBF	Input	\Base\Input	I-75 Master Plan
	EETRIPS	DBF	Input	\Base\Input	I-75 Master Plan
	INTEXT	DBF	Input	\Base\Input	I-75 Master Plan
	PRODRATES	DBF	Parameters	\Parameters	Gainesville 2000
	SPECGEN	DBF	Input	\Base\Input	I-75 Master Plan
	UFData	DBF	Input	\Base\Input	Gainesville 2000
	ZONEDATA	DBF	Input	\Base\Input	MTPO staff
Highway Network	HNET	NET	Input	\Base\Input	I-75 Master Plan
	VFACTORS	CSV	Parameters	\Parameters	Olympus
	SPDCAP	DBF	Parameters	\Parameters	Olympus
	TURN	PEN	Input	\Base\Input	Gainesville 2000
Trip Distribution	FF	DBF	Parameters	\Parameters	Gainesville 2000
Transit Network	AMPNR	DBF	Parameters	\Parameters	Gainesville 2000
	TFACWKLB	FAC	Parameters	\Parameters	Gainesville 2000
	TFACWKPR	FAC	Parameters	\Parameters	Gainesville 2000
	TFACPNR	FAC	Parameters	\Parameters	Gainesville 2000
	TFARES	FAR	Input	\Base\Input	Gainesville 2000
	TRANSPD	DBF	Parameters	\Parameters	Olympus
	TROUTE	LIN	Input	\Base\Input	Gainesville 2000
	TSYS	PTS	Parameters	\Parameters	Gainesville 2000


Gainesville/Alachua County Model
Validated to base year 2007 by Cambridge Systematics, Inc.
Zonal Data
Special Generator
EI Trips
EETarget
Generation
00
1
Generation Summa
EETAB
P and A File
Input Network
Highway Network
00
2
Unloaded Network
Node Coords
Travel Times
Time Skim
P and A File
EETAB
Network File
Turn Penalties
FF (DBF)
Distribution
00
3
Print File
Person Trips
Preload Trips
Preload Net
Pk Hwy LOS
ATL Report
Nodes Coords
Preload Net
Turn Penalties
Transit Routes
Preload LOS
Transit Networks
00
4
Pk Walk Trn Net
Pk Walk LB Paths
Pk Walk EX Paths
Pk Auto Trn Net
Pk Auto BA Paths
Pk Trn LOS
Op Walk Trn Net
Op Walk LB Paths
Op Walk EX Paths
Op Auto Trn Net
Op Auto BA Paths
Op Trn LOS
Person Trips
Pk Transit LOS
OP Transit LOS
Pk Hwy LOS
EETRIPS
Mode Choice
00
5
Mode Summary
Vehicle Trips
Transit Trips
Non-motor Trips
VehTT_UF-Non
Vehicle Trips
Unloaded Network
Pk Walk Trn Net
Pk Walk LB Paths
Pk Walk EX Paths
Transit Trips
Pk Auto Trn Net
Pk Auto BA Paths
Op Walk Trn Net
Op Walk LB Paths
Op Walk EX Paths
Op Auto Trn Net
Op Auto BA Paths
Non-motor Trips
Assignment
00
6
AMwalktransit
AMautotransit
MDwalktransit
MDautotransit
Output Network

Trip Generation Step





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_2007

; =================================================================
; 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
   ;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
ELSEIF (HHTYPE=2); MULTI-FAMILY
   UNITS=   ZI.1.MFDU
   VACRATE= ZI.1.MF_SEA
   ;VACRATE= ZI.1.MF_SEA+ZI.1.MF_VAC
   PERMVACRATE= ZI.1.MF_VAC
   POP=     ZI.1.MFPOP
   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), 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
 SERVEMP=ZI.1.SERVEMP
IF(ZI.1.SERVEMP>ZI.1.UF_EMP)
```

```
 SERVEMP=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
 SERVEMP =  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)
 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]
```

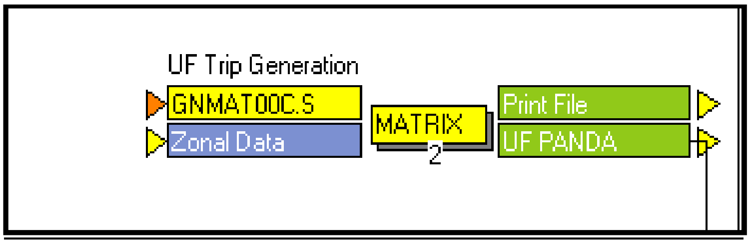
```
P[6]=A[6]
P[7]=A[7]
P[8]=A[8]

;
.........................................................................
; 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

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

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
```

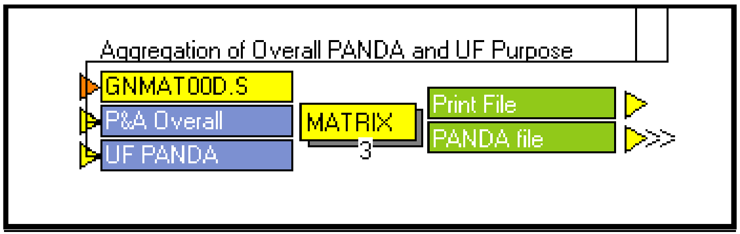
```
PRINT LIST="  Internal Person Trips per Total Occupied Dwelling Units =
",ITPTOC(5.3C), PRINTO=1
PRINT    LIST="        Internal    Person    Trips    per    Employee    =
",ITPEMP(5.3C), PRINTO=1


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_2007
FILEO RECO[1] = "{SCENARIO_DIR}\output\UFPANDA.dbf",
               FIELDS=Z,HBUP,HBUA,HDORMUP,HDORMUA,STUPCT,nocarpct,wcarpct

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)
; 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
```

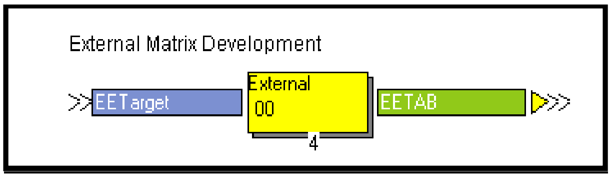
```
ENDRUN
```

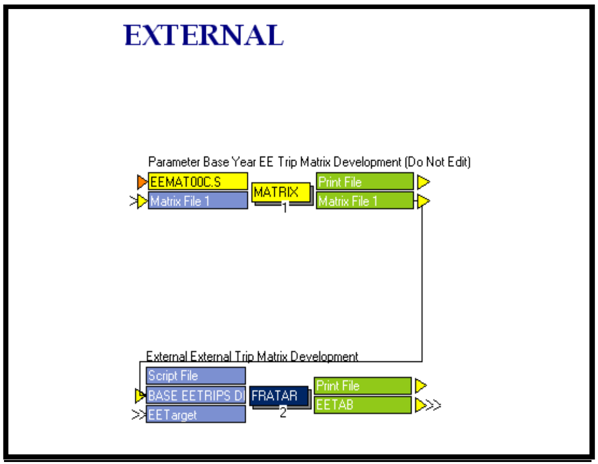


GNMAT00D.S

```
; 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,L
DTKIEA,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
```

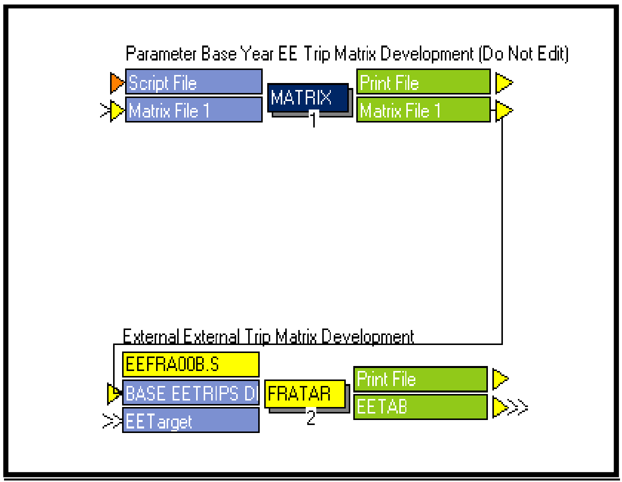
```
ENDRUN
```





EEMAT00C.S

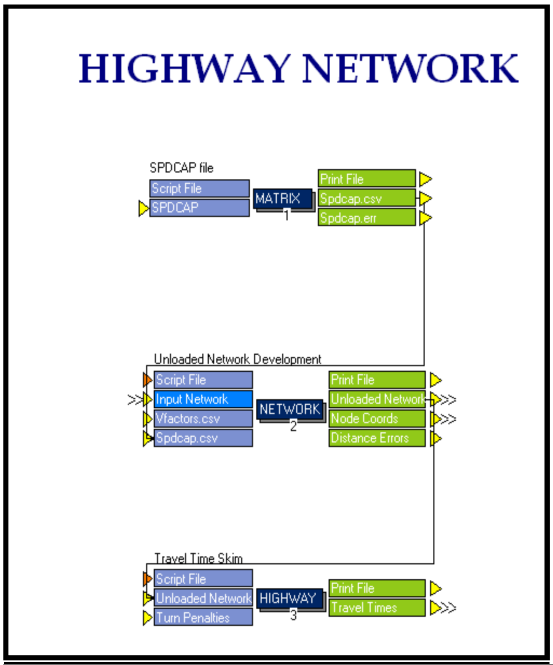
```
; Do not change filenames or add or remove FILEI/FILEO statements using
an editor. Use Cube/Application Manager.
RUN         PGM=MATRIX         PRNFILE="{SCENARIO_DIR}\output\EEMAT00c.PRN"
MSG='Parameter Base Year EE Trip Matrix Development (Do Not Edit)'
FILEO MATO[1] = "{CATALOG_DIR}\PARAMETERS\BASEYEAR_EETRIPS_DIST.MAT",
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
```

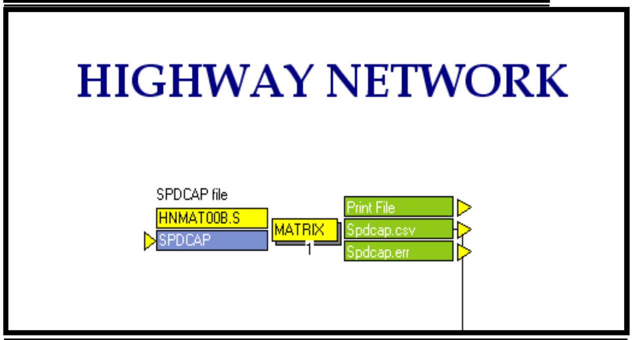


EEFRA00B.S

```
; 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, name=EETRIPS

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
ACOMP=1,PCOMP=1
MARGINS=1

ENDRUN
```

Highway Network Step



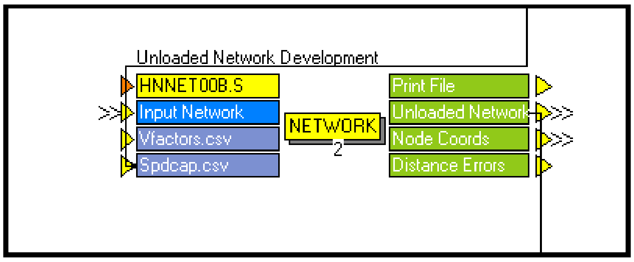


<u>HNMAT00B.S</u>

```
; 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
```

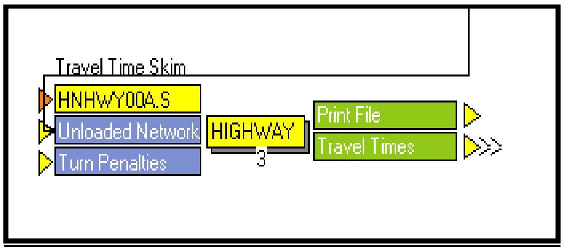


HNNET00B.S

```
; 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 (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
```

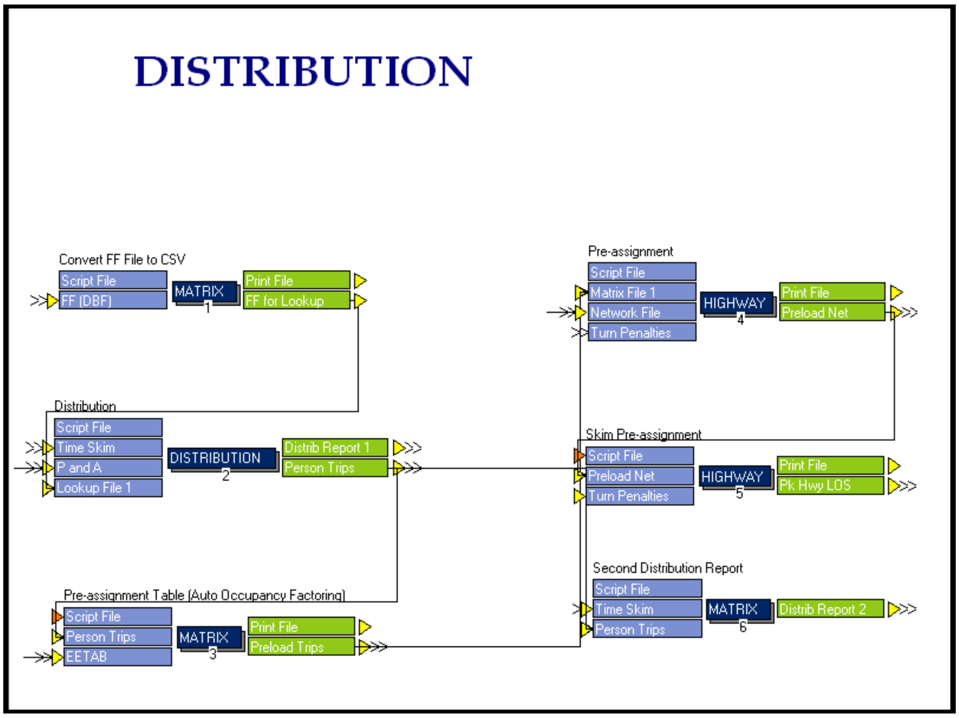


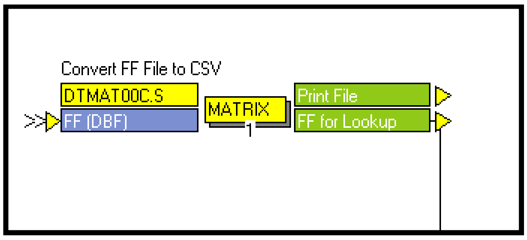
HNHWY00A.S

```
; 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=10-19,49,70-99) ADDTOGROUP=1 ; no walk on freeways, etc.
ENDPROCESS


PROCESS PHASE=ILOOP
  PATHLOAD PATH=LI.TIME,
      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,        MW[99]=PATHTRACE(LI.DISTANCE),
EXCLUDEGROUP=1
  MW[99][I]=ROWMIN(99)/2
ENDPROCESS


PROCESS PHASE=ADJUST


ENDPROCESS

ENDRUN
```

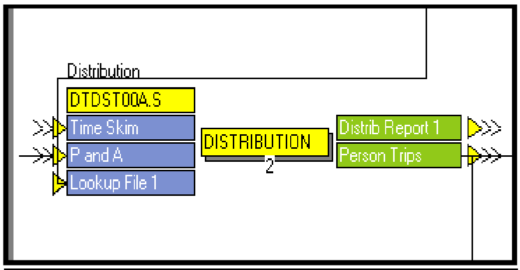
Trip Distribution Step





<u>DTMAT00C.S</u>

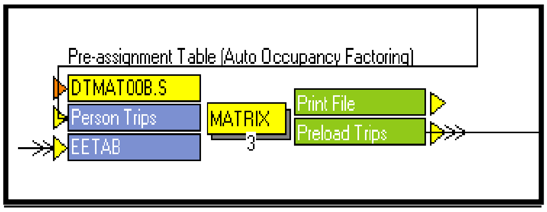
```
; Do not change filenames or add or remove FILEI/FILEO statements using
an editor. Use Cube/Application Manager.
RUN PGM=MATRIX PRNFILE="{SCENARIO_DIR}\Output\DTMAT00A.PRN" MSG='Convert
FF File to CSV'
FILEO PRINTO[1] = "{SCENARIO_DIR}\OUTPUT\FF.CSV"
FILEI RECI = "{CATALOG_DIR}\PARAMETERS\FF.DBF"

print                                                              csv=t,
list=ri.time,ri.hbwff,ri.hbshff,ri.hbsrff,ri.hboff,ri.nhbff,ri.tk4ff,ri.t
ksglff,ri.tktrlrff,

ri.sovieff,ri.hovieff,ri.tkltieff,ri.tkhtieff,ri.hbuff,ri.hdormuff,
printo=1

ENDRUN
```



DTDST00A.S

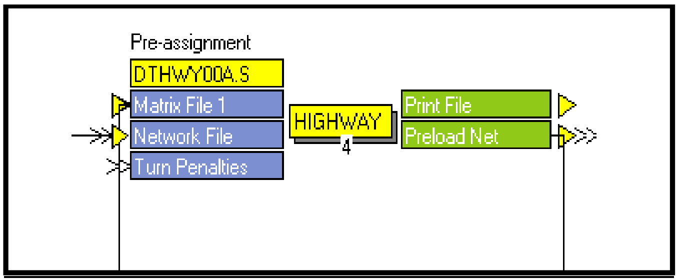
```
; Do not change filenames or add or remove FILEI/FILEO statements using
an editor. Use Cube/Application Manager.
RUN      PGM=DISTRIBUTION      PRNFILE="{SCENARIO_DIR}\output\DISTRIB.PRN"
MSG='Distribution'
FILEO MATO[1] = "{SCENARIO_DIR}\Output\PTRIPS.MAT",
 MO=1-14,
NAME=HBW,HBSH,HBSR,HBO,NHB,TRUCK4,TRUCKSU,TRUCKTRLR,SOVIE,HOVIE,TRUCKLDIE
,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=0-60-1.0, TITLE='HBW TLFD'
FREQUENCY BASEMW=50,VALUEMW= 2,RANGE=0-60-1.0, TITLE='HBSH TLFD'
FREQUENCY BASEMW=50,VALUEMW= 3,RANGE=0-60-1.0, TITLE='HBSR TLFD'
FREQUENCY BASEMW=50,VALUEMW= 4,RANGE=0-60-1.0, TITLE='HBO TLFD'
FREQUENCY BASEMW=50,VALUEMW= 5,RANGE=0-60-1.0, TITLE='NHB TLFD'
FREQUENCY BASEMW=50,VALUEMW= 6,RANGE=0-60-1.0, TITLE='TK4 TLFD'
FREQUENCY BASEMW=50,VALUEMW= 7,RANGE=0-60-1.0, TITLE='SGLUNIT TLFD'
FREQUENCY BASEMW=50,VALUEMW= 8,RANGE=0-60-1.0, TITLE='TRKTLR TLFD'
FREQUENCY BASEMW=50,VALUEMW= 9,RANGE=0-60-1.0, TITLE='SOVIE TLFD'
FREQUENCY BASEMW=50,VALUEMW=10,RANGE=0-60-1.0, TITLE='HOVIE TLFD'
FREQUENCY BASEMW=50,VALUEMW=11,RANGE=0-60-1.0, TITLE='LDTKIE TLFD'
FREQUENCY BASEMW=50,VALUEMW=12,RANGE=0-60-1.0, TITLE='HDTKIE TLFD'
FREQUENCY BASEMW=50,VALUEMW=13,RANGE=0-60-1.0, TITLE='HBU TLFD'
FREQUENCY BASEMW=50,VALUEMW=14,RANGE=0-60-1.0, TITLE='HDORMU TLFD'


ENDRUN
```



DTMAT00B.S

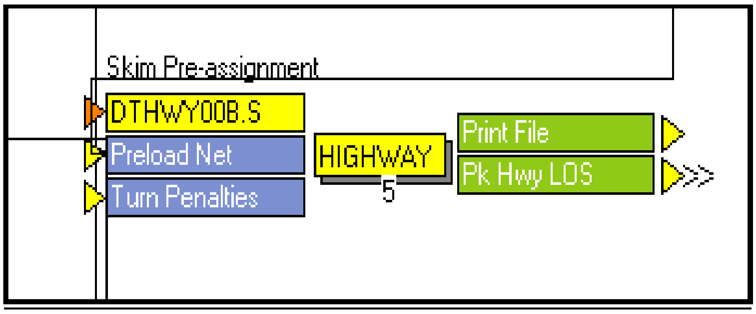
```
; Do not change filenames or add or remove FILEI/FILEO statements using
an editor. Use Cube/Application Manager.
RUN  PGM=MATRIX  PRNFILE="{SCENARIO_DIR}\Output\DTMAT00B.PRN"  MSG='Pre-
assignment Table (Auto Occupancy Factoring)'
FILEI MATI[2] = "{SCENARIO_DIR}\output\EETAB.MAT"
FILEI MATI[1] = "{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

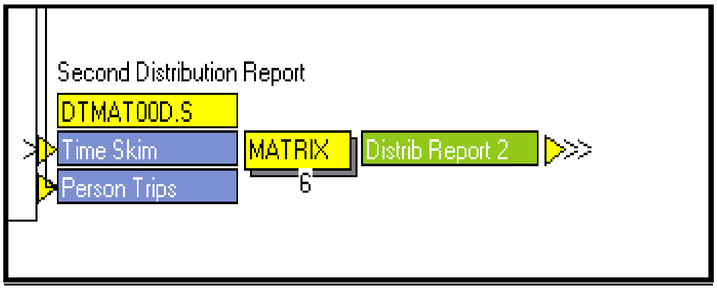
```
; Do not change filenames or add or remove FILEI/FILEO statements using
an editor. Use Cube/Application Manager.
RUN  PGM=HIGHWAY  PRNFILE="{SCENARIO_DIR}\Output\DTHWY00A.PRN"  MSG='Pre-
assignment'
FILEI NETI = "{SCENARIO_DIR}\OUTPUT\UNLOADED.NET"
FILEO NETO = "{SCENARIO_DIR}\OUTPUT\PRELOAD.NET"
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

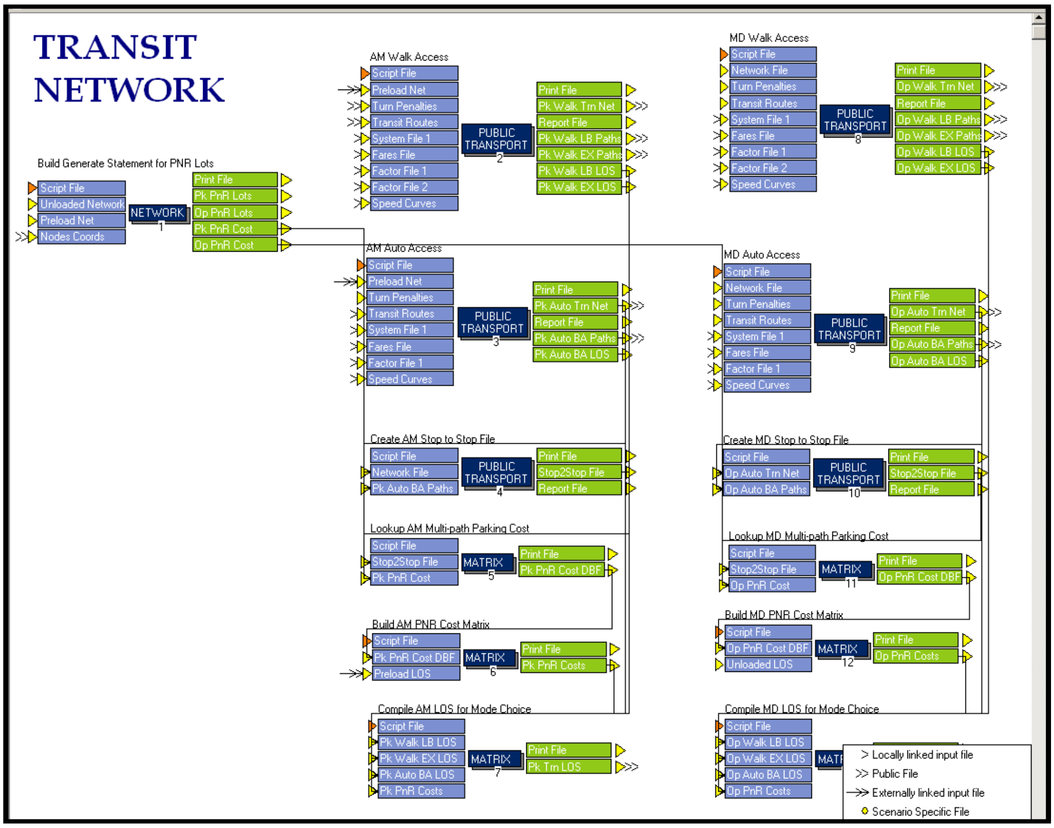
```
; Do not change filenames or add or remove FILEI/FILEO statements using
an editor. Use Cube/Application Manager.
RUN  PGM=HIGHWAY  PRNFILE="{SCENARIO_DIR}\Output\DTHWY00B.PRN"  MSG='Skim
Pre-assignment'
FILEI NETI = "{SCENARIO_DIR}\OUTPUT\PRELOAD.NET"
FILEO MATO[1] = "{SCENARIO_DIR}\OUTPUT\RHSKIMS.MAT",
 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)
ENDPROCESS


PROCESS PHASE=ILOOP
  PATHLOAD PATH=LI.TIME,
      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
```

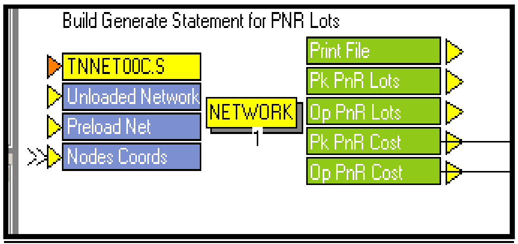


DTMAT00D.S

```
; 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



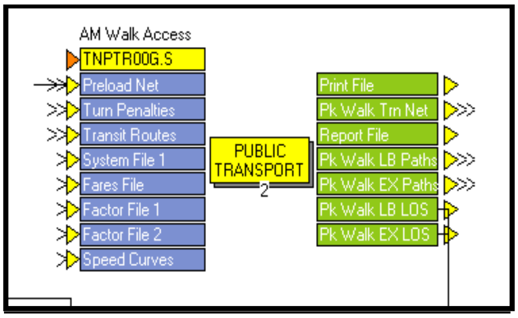


TNNET00C.S

```
; 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*",pn
rsvcarea(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*",pn
rsvcarea(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
```

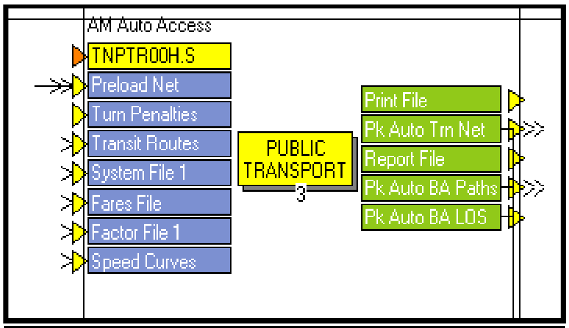


TNPTR00G.S

```
; 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=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_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
```

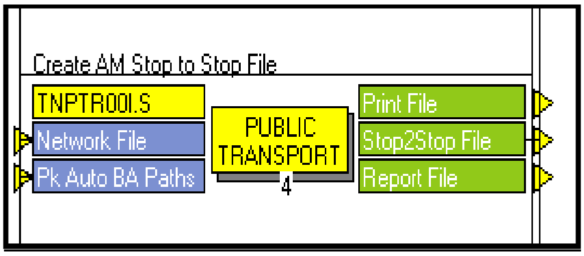


TNPTR00H.S

```
; 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=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_1>0) LW.SPEED=60*LI.DISTANCE/LI.TIME_1
```

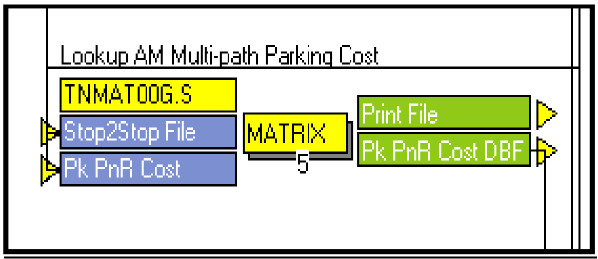
```
   LOOKUP, NAME=CURVES, LOOKUP[1]=1, RESULT=2, LOOKUP[2]=1, RESULT=3,
LOOKUP[3]=1, RESULT=4,
           INTERPOLATE=Y, LOOKUPI=1
   IF (LI.FTYPE=10-19,80-99) ; FREE FLOW CONDITIONS FOR ALL
       LW.LBCURVE=1,LW.EBCURVE=1,LW.RLCURVE=1
   ELSEIF (LI.FTYPE=20-79&LI.ATYPE=10-19) ; BUSSES HITTING RESISTANCE,
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=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
```



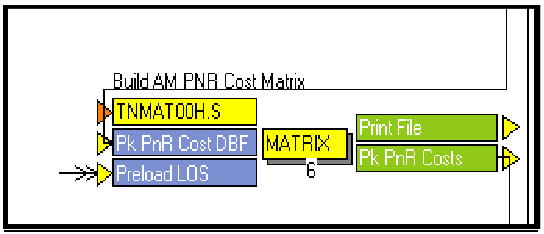
TNPTR00I.S

```
; 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
```



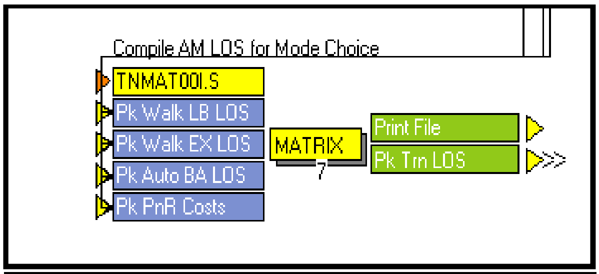
TNMAT00G.S

```
; 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

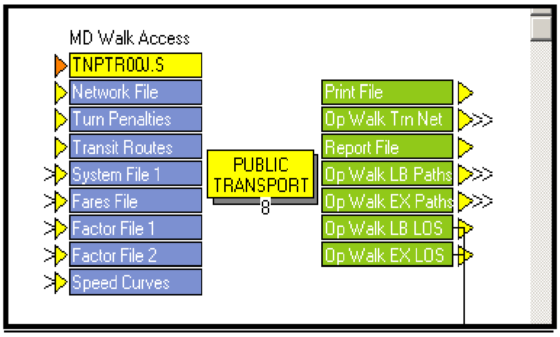
```
; Script for program MATRIX in file "C:\FSUTMS\DISTRICT2\ALACHUA\TNMAT00C.S"
; Do not change filenames or add or remove FILEI/FILEO statements using an editor. Use Cube/Application Manager.
RUN PGM=MATRIX PRNFILE="{SCENARIO_DIR}\output\TNMAT00C.PRN" MSG='Build AM PNR Cost Matrix'
FILEI MATI[2] = "{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
```

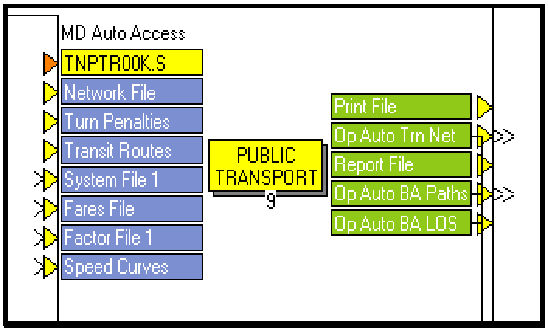


TNPTR00J.S

```
;     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
;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
```

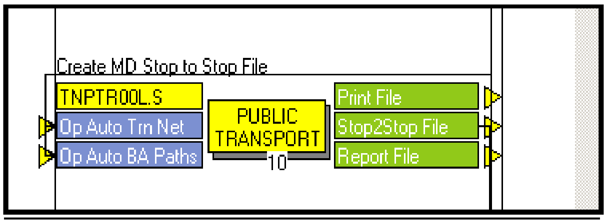


<u>TNPTR00K.S</u>

```
; 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
```

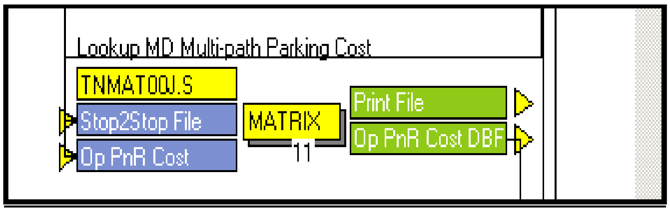
```
   LOOKUP,  NAME=CURVES,  LOOKUP[1]=1,  RESULT=2,  LOOKUP[2]=1,  RESULT=3,
LOOKUP[3]=1, RESULT=4,
           INTERPOLATE=Y, LOOKUPI=1
   IF (LI.FTYPE=10-19,80-99) ; FREE FLOW CONDITIONS FOR ALL
       LW.LBCURVE=1,LW.EBCURVE=1,LW.RLCURVE=1
   ELSEIF  (LI.FTYPE=20-79&LI.ATYPE=10-19)  ;  BUSSES  HITTING  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
```



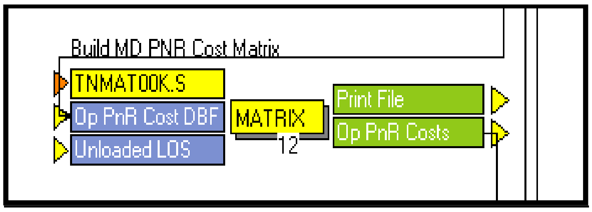
TNPTR00L.S

```
; 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
```



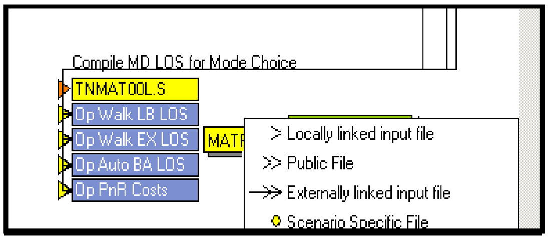
TNMAT00J.S

```
; 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
```

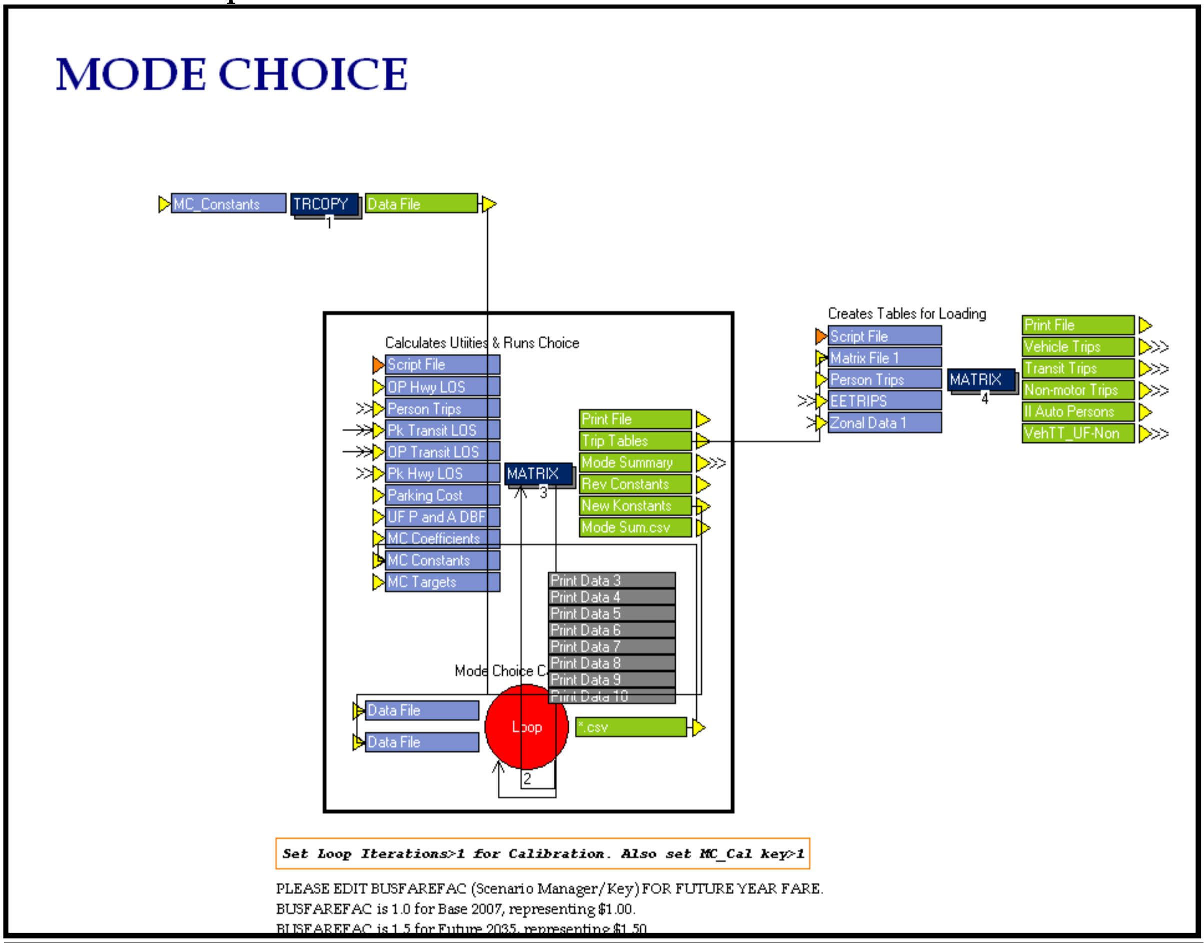


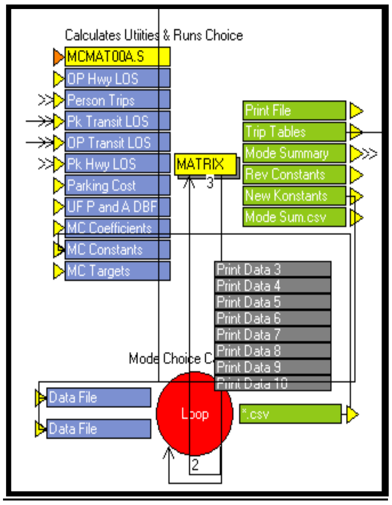
<u>TNMAT00L.S</u>

```
; 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_2007
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);/{NESTCTRANS
IT}

  ;   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)*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)*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)*HBOCCST
      MW[15]=0 ; UF fare free
  ;   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)*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=8.0C, 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=8.0C, 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=8.0C,  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),t
2_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),t
2_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),t
2_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),t
2_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),t
2_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),t
2_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),t
2_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),t
2_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,K1_WC_DA,K1_ST_DA,K2_NC_DA,K2_WC_DA,K2_ST_DA,K3_NC_DA,K4_NC_
DA,K5_NC_DA, PRINTO=2
PRINT                             CSV=T,                            LIST=
2.0,K1_NC_CP,K1_WC_CP,K1_ST_CP,K2_NC_CP,K2_WC_CP,K2_ST_CP,K3_NC_CP,K4_NC_
CP,K5_NC_CP, PRINTO=2
PRINT                             CSV=T,                            LIST=
3.0,K1_NC_CX,K1_WC_CX,K1_ST_CX,K2_NC_CX,K2_WC_CX,K2_ST_CX,K3_NC_CX,K4_NC_
CX,K5_NC_CX, PRINTO=2
PRINT                             CSV=T,                            LIST=
4.0,K1_NC_WB,K1_WC_WB,K1_ST_WB,K2_NC_WB,K2_WC_WB,K2_ST_WB,K3_NC_WB,K4_NC_
WB,K5_NC_WB, PRINTO=2
PRINT                             CSV=T,                            LIST=
5.0,K1_NC_WX,K1_WC_WX,K1_ST_WX,K2_NC_WX,K2_WC_WX,K2_ST_WX,K3_NC_WX,K4_NC_
WX,K5_NC_WX, PRINTO=2
PRINT                             CSV=T,                            LIST=
6.0,K1_NC_BA,K1_WC_BA,K1_ST_BA,K2_NC_BA,K2_WC_BA,K2_ST_BA,K3_NC_BA,K4_NC_
BA,K5_NC_BA, PRINTO=2
PRINT                             CSV=T,                            LIST=
7.0,K1_NC_WK,K1_WC_WK,K1_ST_WK,K2_NC_WK,K2_WC_WK,K2_ST_WK,K3_NC_WK,K4_NC_
WK,K5_NC_WK, PRINTO=2
PRINT                             CSV=T,                            LIST=
8.0,K1_NC_BK,K1_WC_BK,K1_ST_BK,K2_NC_BK,K2_WC_BK,K2_ST_BK,K3_NC_BK,K4_NC_
BK,K5_NC_BK, 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
/*
```

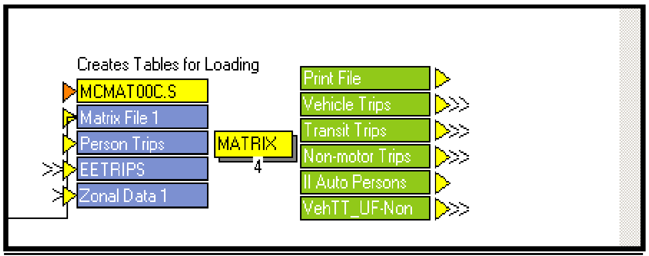
```
NESTMOTOR    = 1.0
NESTTRANSIT  = 1.0
NESTNONMOTOR = 1.0
*/
LCP=Ln(MW402/t1_NC_CP)*NESTMOTOR
LCX=Ln(MW403/t1_NC_CX)*NESTMOTOR
LWB=Ln(MW404/t1_NC_WB)*NESTTRANSIT
LWK=Ln(MW407/t1_NC_WK)*NESTNONMOTOR
LBK=Ln(MW408/t1_NC_BK)*NESTNONMOTOR

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_W
B)+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_W
B)+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_W
B)+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_W
B)+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_W
B)+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_W
B)+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
```

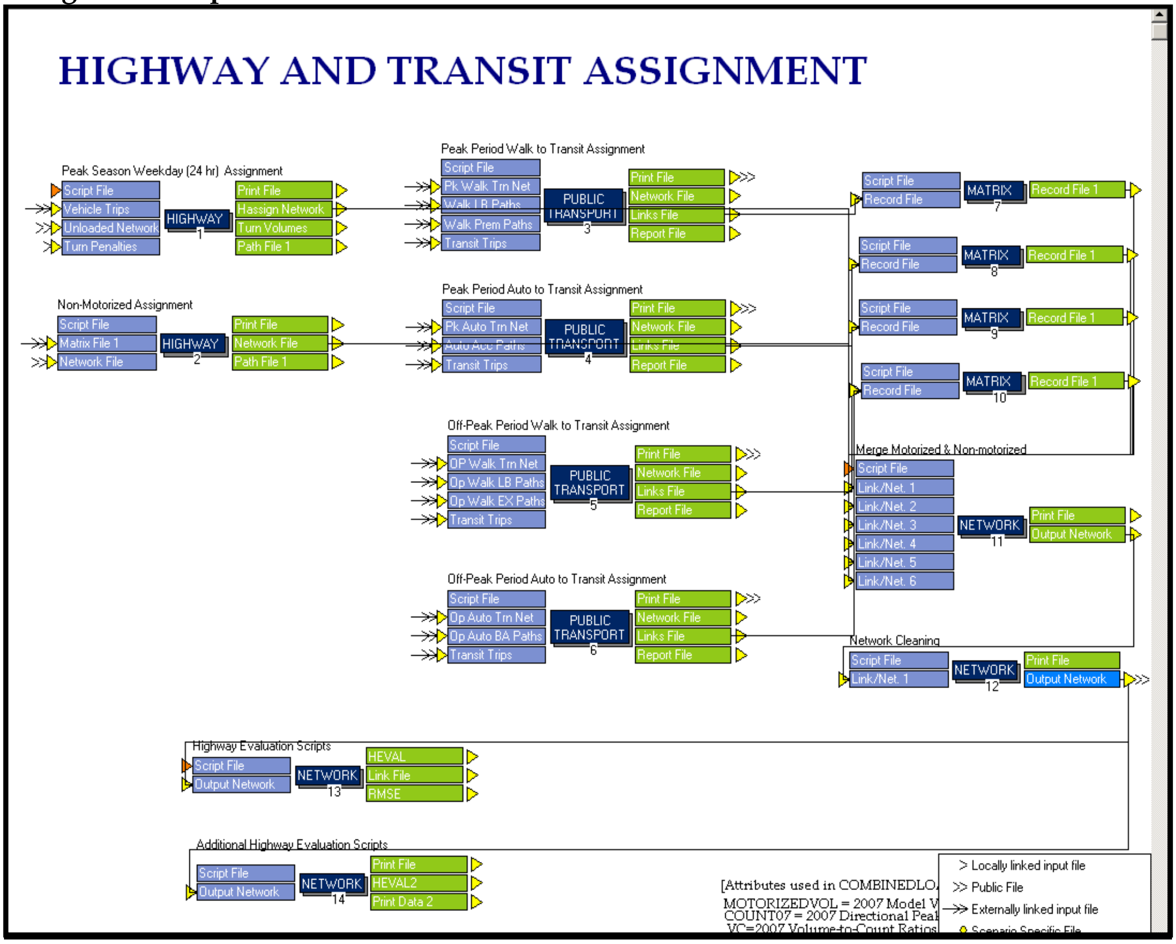


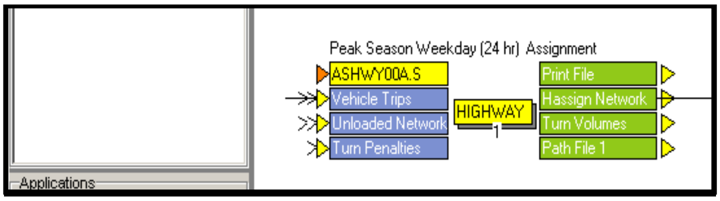
MCMAT00C.S

```
; 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_2007
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 ; 0.5 added by KDK
; 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

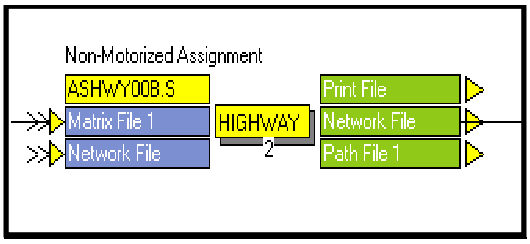




ASHWY00A.S

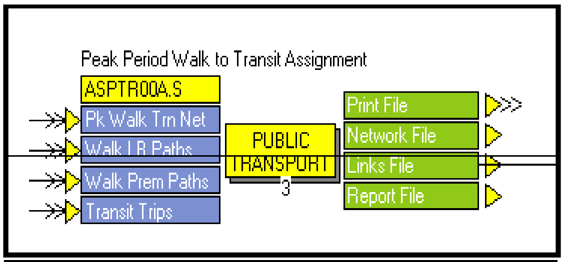
```
; Do not change filenames or add or remove FILEI/FILEO statements using
an editor. Use Cube/Application Manager.
RUN  PGM=HIGHWAY  PRNFILE="{SCENARIO_DIR}\output\HASSIGN.PRN"  MSG='Peak
Season Weekday (24 hr)  Assignment'
FILEI NETI = "{SCENARIO_DIR}\output\UNLOADED.NET"
FILEO TURNVOLO = "{SCENARIO_DIR}\output\TURNVOL.BIN",
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
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.UF_light
  MW[2]=MI.1.UF_heavy*{PCE_HT}
  MW[3]=MI.1.Non_light
  MW[4]=MI.1.Non_heavy*{PCE_HT}
  MW[5]=MI.1.SelZones_light
  MW[6]=MI.1.SelZones_heavy*{PCE_HT}
  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]=M
W[6], PENI=1, EXCLUDEGROUP=1,
  PATHO=1, ALLJ=T, INCLUDECOST=F, NAME=ALLTRIPS
ENDPROCESS


PROCESS PHASE=ADJUST
  V = VOL[1] + VOL[2]+ VOL[3] + VOL[4]
  FUNCTION        TC[1]=T0*(1+LW.BPRCOEFFICIENT*(V/C)^LW.BPREXPONENT)   ;
congested time equation, no toll model in place
ENDPROCESS

ENDRUN
```



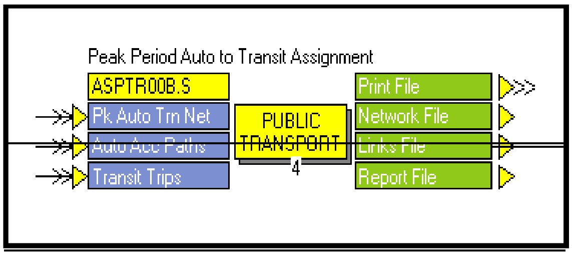
ASHWY00B.S

```
; 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
```



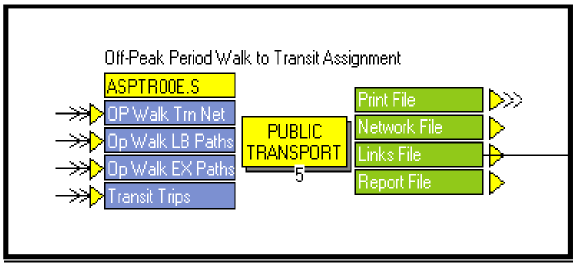
ASPTR00A.S

```
; 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
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
```



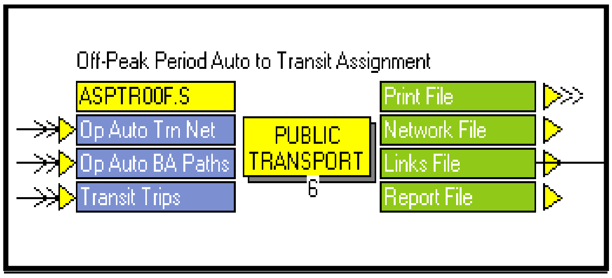
ASPTR00B.S

```
; 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 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
```



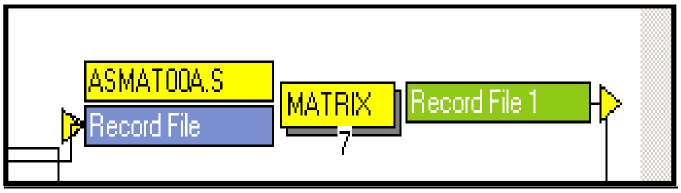
ASPTR00E.S

```
; 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
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
```



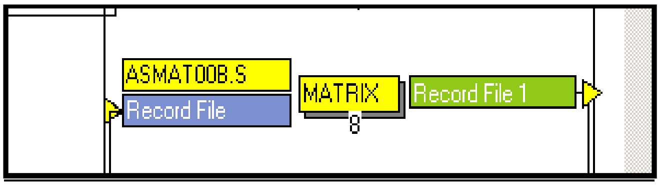
ASPTR00F.S

```
; 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
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
```



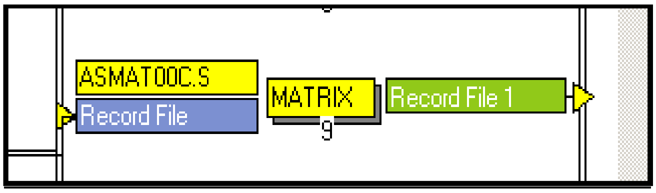
ASMAT00A.S

```
; Do not change filenames or add or remove FILEI/FILEO statements using
an editor. Use Cube/Application Manager.
RUN PGM=MATRIX
FILEO RECO[1] = "{SCENARIO_DIR}\output\LL1.DBF",
  FIELDS=A,B,MODE,NAME,DIST,TIME,SEQ,CNT,HEADWAY_1,VOL
FILEI RECI = "{SCENARIO_DIR}\output\TLOD1.DBF"
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
```



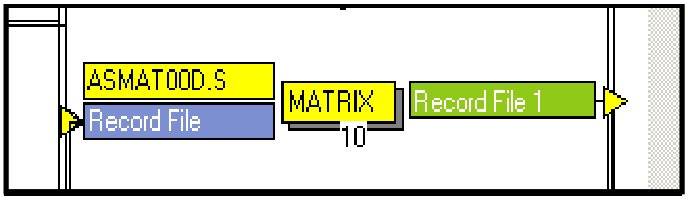
ASMAT00B.S

```
; 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
```



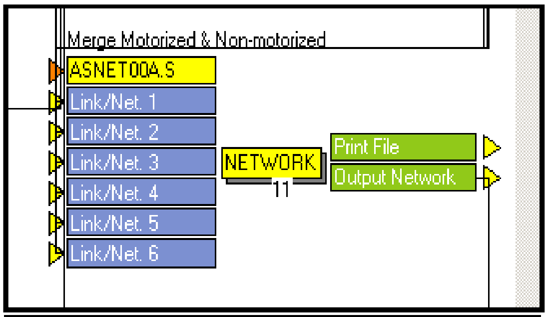
ASMAT00C.S

```
; 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

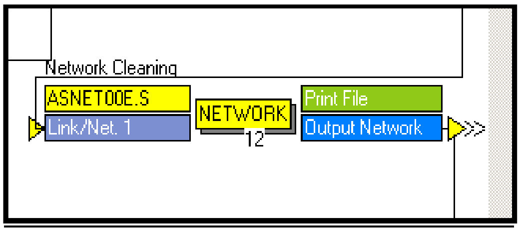
```
; Do not change filenames or add or remove FILEI/FILEO statements using
an editor. Use Cube/Application Manager.
RUN PGM=MATRIX
FILEI RECI = "{SCENARIO_DIR}\output\TLOD4.DBF"
FILEO RECO[1] = "{SCENARIO_DIR}\output\LL4.DBF",
  FIELDS=A,B,MODE,NAME,DIST,TIME,SEQ,CNT,HEADWAY_2,VOL
vtot=vtot+ri.VOL
if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

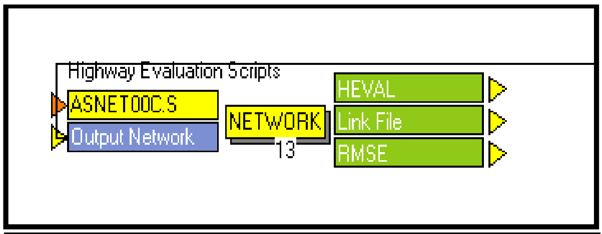
```
; Do not change filenames or add or remove FILEI/FILEO statements using
an editor. Use Cube/Application Manager.
RUN  PGM=NETWORK  PRNFILE="{SCENARIO_DIR}\output\ASNET00A.PRN"  MSG='Merge
Motorized & Non-motorized'
FILEO NETO = "{SCENARIO_DIR}\output\COMB_TEMP.NET"
;FILEO PRINTO[1] = "{SCENARIO_DIR}\output\UFOUCH.PRN"
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}
  _NON_LIGHT=LI.1.V3_1
  _NON_HEAVY=LI.1.V4_1/{PCE_HT}
  _SELZONE_LIGHT=LI.1.V5_1
  _SELZONE_HEAVY=LI.1.V6_1/{PCE_HT}
  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

  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(COUNT07 > 0)
  VC=MOTORIZEDVOL/COUNT07
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="D:\FSUTMS\D2\GAINESVILLE_2007_2035\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"



ENDRUN
```



ASNET00C.S

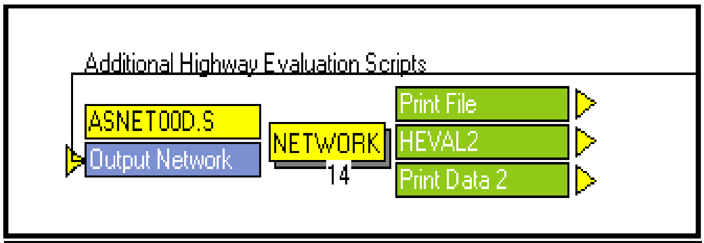
```
; Do not change filenames or add or remove FILEI/FILEO statements using
an editor. Use Cube/Application Manager.
RUN      PGM=NETWORK      PRNFILE="{SCENARIO_DIR}\output\HEVAl_Daily.PRN"
MSG='Highway Evaluation Scripts'
FILEO LINKO = "{SCENARIO_DIR}\output\Daily_Links.DBF"
FILEI LINKI[1] = "{SCENARIO_DIR}\output\COMBINEDLOADED.NET"
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.COUNT07
CAP         =LI.1.CAPACITY*LI.1.LANES/LI.1.CONFAC


;

   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
   _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
     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 (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
   _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
CROSSTAB  VAR= _A_VMT_ALL, form=14.0c,
    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

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[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.COUNT07                          ;Traffic Count
_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


;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

   _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


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), 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

ENDPROCESS

ENDRUN
```

Appendix F: Input and Output Network Format

Input Network Format (HNET20{YEAR}.NET)

Note: {YEAR} represents a scenario year in the last two digits. The file name will be HNET2007.NET if it is base year 2007 scenario or HNET2035.NET if it is 2035 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=twoway, 1=oneway)

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 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.

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

■ 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 – 2007 Volume-to-Count Ratio (VC=MOTORIZEDVOL/COUNT07) This is only available in the base year 2007 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

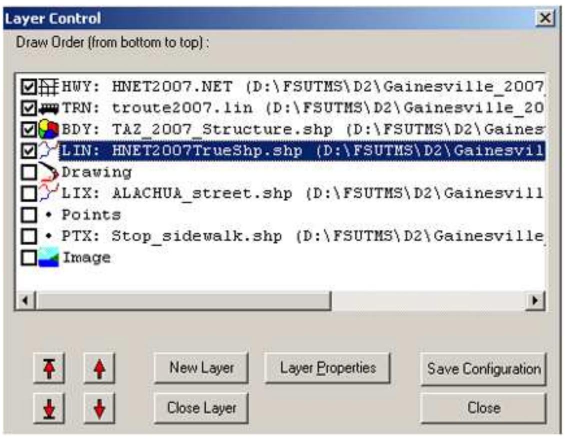
BK_TIME – Bicycle travel time in minutes

Notes Regarding True Shepe Display of Networks

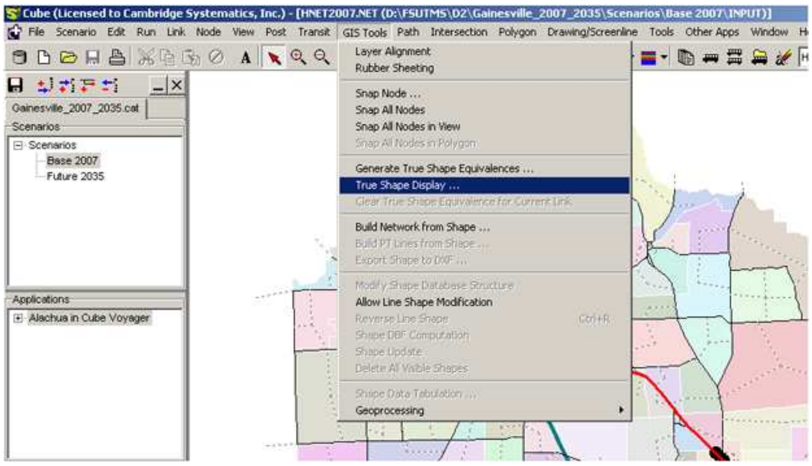
HNET20{YEAR}.NET and COMBINEDLOADED.NET in the Gainesville MTPO 2007 should be applied with True Shapefile Display in Cube software to be shown with curved line shape. Please use True Shape polyline GIS shapefile which is available in model data in the following file location.

…\Gainesville_2007_2035\Media\Street\HNET2007TrueShp.shp

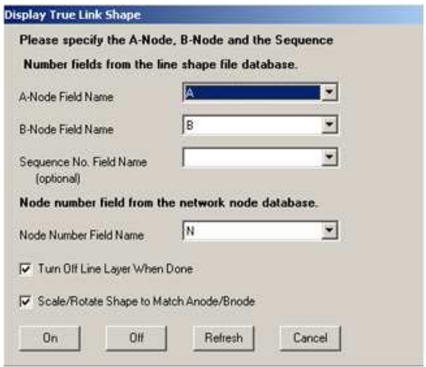
Step 1. Make sure that True Shape GIS shapefile is correctly navigated and overlaid underneath the input or output network, as shown in the next screenshot



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 Ok.



Now, you should have a network displayed with curved shape instead of straight line shape.

Appendix G: Glossary and Abbreviations

GLOSSARY OF TERMS

(Source FSUTMS 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. FSUTMS Comprehensive Modeling Workshop – Glossary of Terms Page 14

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 afunction 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. FSUTMS Comprehensive Modeling Workshop – Glossary of Terms Page 15

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) - FSUTMS Comprehensive Modeling Workshop – Glossary of Terms Page 16 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 for 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.
FSUTMS Comprehensive Modeling Workshop – Glossary of Terms Page 18

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 Comprehensive Modeling Workshop – Glossary of Terms Page 19

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 zone-to-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. FSUTMS Comprehensive Modeling Workshop – Glossary of Terms Page 20

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 Directional traffic split

D30 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

FIHS Florida Intrastate Highway System

FM Financial Management

FPI Financial Project Identifier

FSUTMS Florida Standard Urban Transportation Model Structure computer program

HCM Highway Capacity Manual

K30 Ratio of DHV to AADT 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 FPI)

Additional Resources and/or Recommended Readings

FSUTMS New Standards and Enhancements – A User Oriented Approach, A Florida Model Task Force White Paper, Florida Department of Transportation, Systems Planning Office, January 2006. http://www.fsutmsonline.net/images/uploads/mtf-files/whitepaper.pdf

FSUTMS Powered by CUBE/Voyager Data Dictionary, Florida Department of Transportation, Systems Planning Office, December 2005. http://www.fsutmsonline.net/images/uploads/mtf-files/datadictionary.pdf

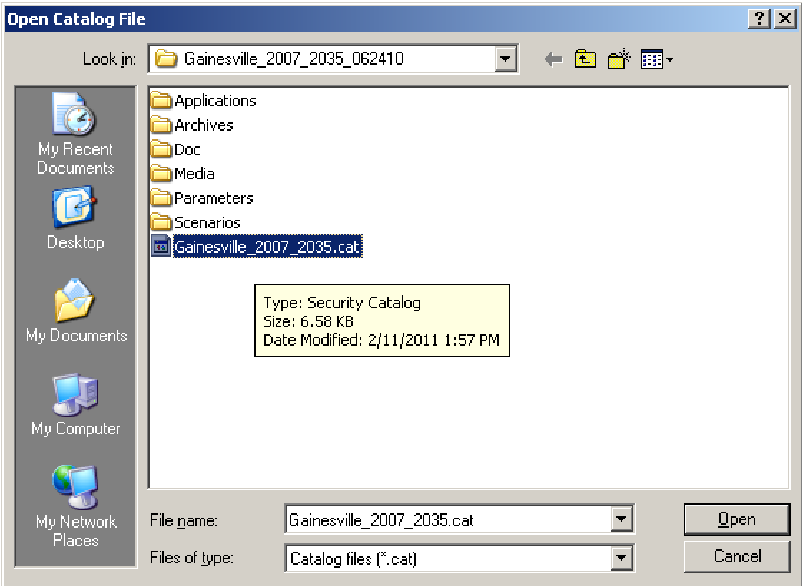
FSUTMS Transit Model Application Guide, Florida Department of Transportation, Systems Planning Office, October 2008. http://www.fsutmsonline.net/images/uploads/reports/TM_ApplicationGuide.pdf

Appendix H: Scenario Manager / Running Model

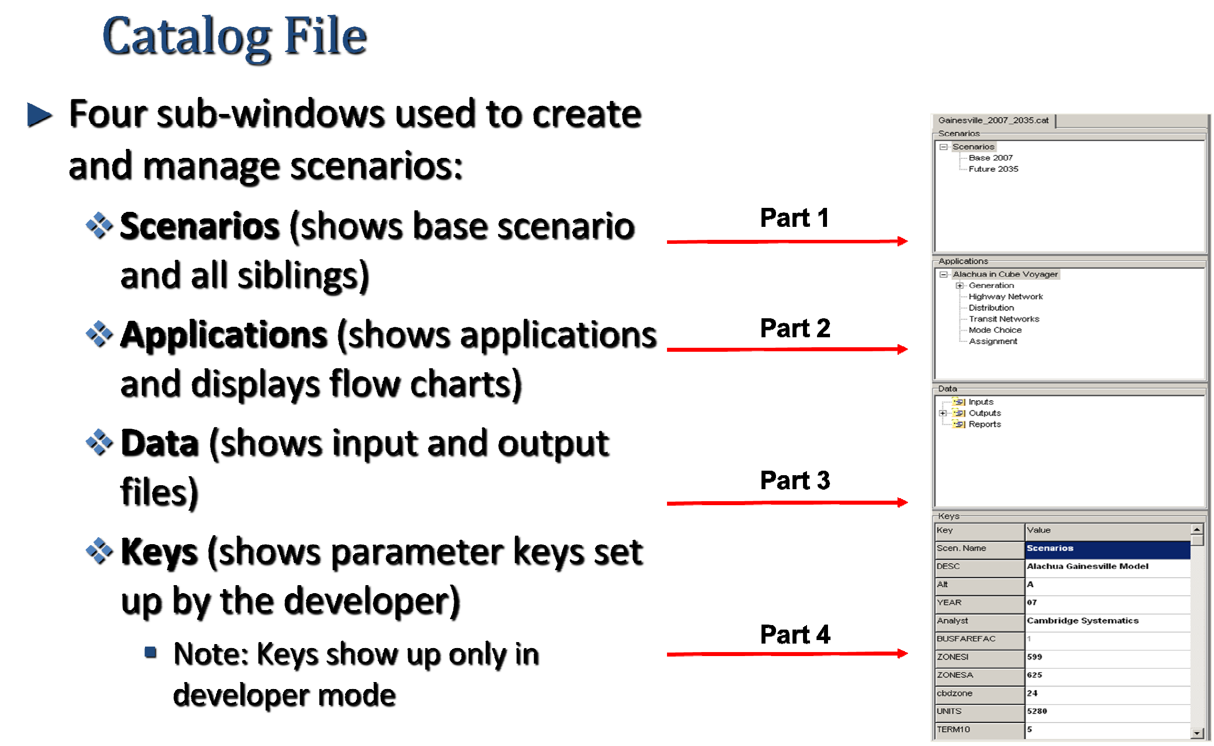
1. To run the Alachua County model, you first have to open up the Cube Application window by navigating to it or clicking the Cube desktop icon.

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.



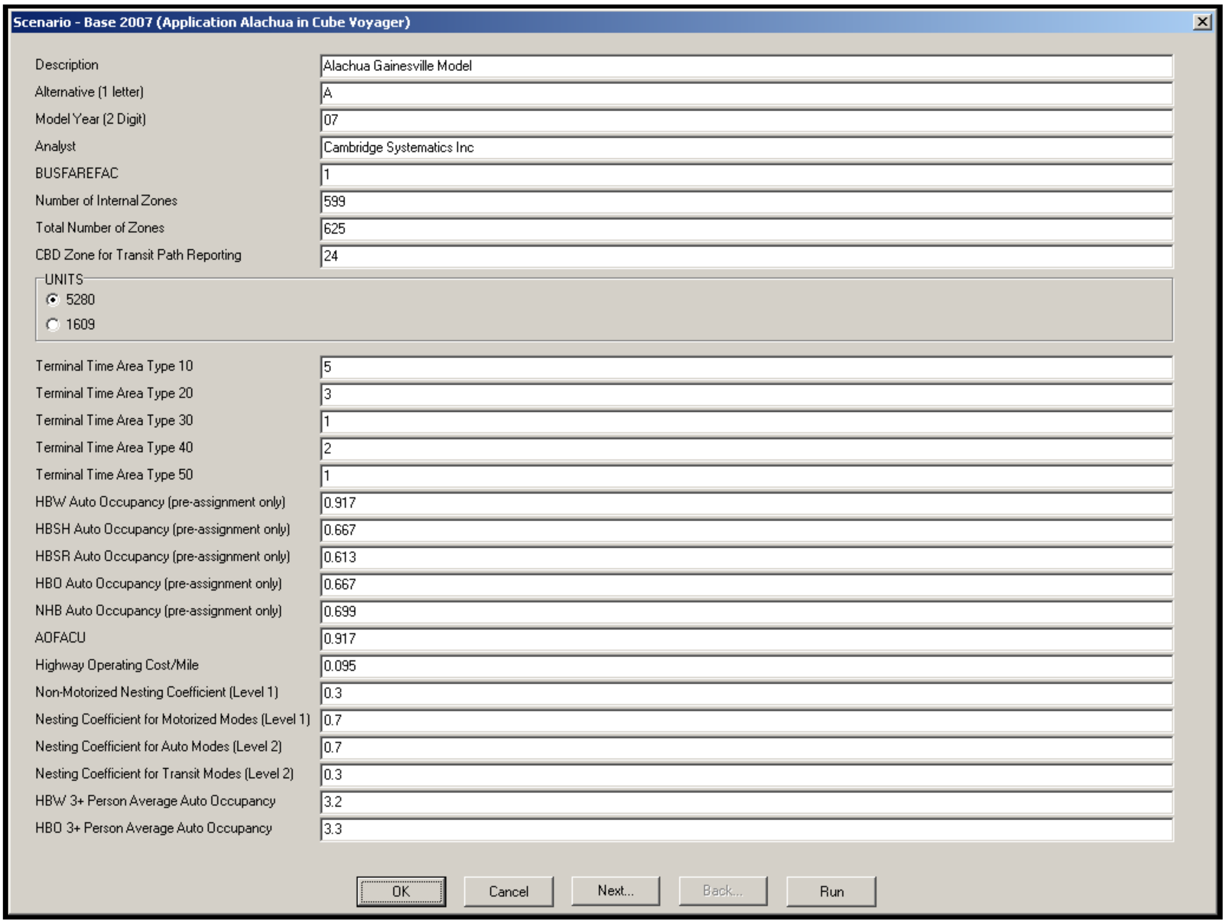
4. This brings up the Gaineville modelapplication window.



The dialog box above is referenced from the FSUTMS Comprehensive Workshop and illustrates the various parts of the Cube catalog file.

5. To run a particular scenario, for example the Base Year, you would go to the Part 1 section, expand the folder beside Scenarios and choose **Base 2007.**

When chosen, the Scenario dialog box is displayed as illustrated below.



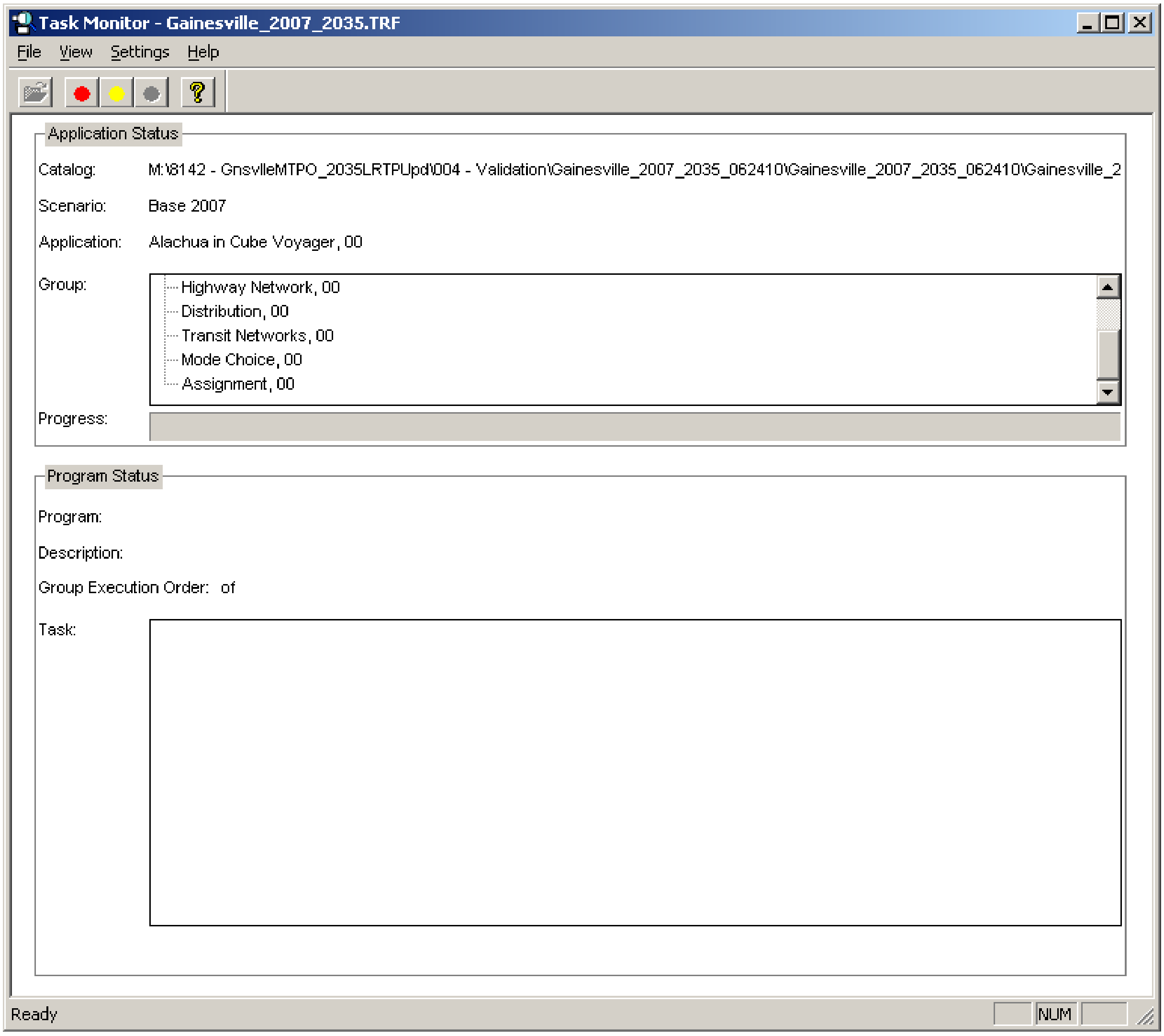
This dialog box allows you to change model runtime options and set model parameters. These parameters are stores as Cube Catalog Keys which are effectively variables which are references in the model script during the relevant model processing phase.The Catalog Key values can also be observed in Part 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 general mode.

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 FSUTMS models, you may refer to the content in the FSUTMS Comprehensive Modeling Workshops held periodically throughout the year in various locations across the state.


Task Monitor - Gainesville_2007_2035.TRF
File View Settings Help
Application Status
Catalog: M:\8142 - GnsvlleMTPO_2035LRTPUpd\004 - Validation\Gainesville_2007_2035_062410\Gainesville_2007_2035_062410\Gainesville_2
Scenario: Base 2007
Application: Alachua in Cube Voyager, 00
Group:
Highway Network, 00
Distribution, 00
Transit Networks, 00
Mode Choice, 00
Assignment, 00
Progress:
Program Status
Program:
Description:
Group Execution Order: of
Task:
Ready
NUM